SEPTEMBER 1990 £2.75

Commodore DISK·USER

BANKING C128

Don't go around in circles keeping track of your money

- Adventure Helpline
- Exploring the 1541
- Inside Basic Memory

9 770953 061007

NOW IS THE TIME
TO CATCH UP ON ISSUES YOU HAVE MISSED

VOL 2 No.5 JUL/AUG '89
EONT FACTORY - Create your own characters.
HIRES DEMO KIT - Add music to your favourite picture.
ANIMATOR - Get those sprites moving
BORDER MESSAGE SCROLL - Say what you like along the bottom of the screen.
TYPIT-128 - Create professional text layout on your C128
SCREEN COPIES UTILITY - Download your favourite screens, including CDU paint files.
VIDI-BASIC - Graphic based extension to Basic.
64 NEWS DESK - Become a C64 reporter.

VOL 2 No.6 SEP/OCT '89
MICKMON - An extensive M/C monitor.
SCRAPBOOK - Collectors and hobbyists database.
CELLRATOR - Enter the caves if you dare.
RAINBOW CHASER - Rainbows means points in this unusual game.
HIDDEN GRAPHICS - Utilise those graphic screens.
FORTRESS - Save the world! Yet again.
DISK HUNTER - Keep tabs on your disk library.
SUPERFILE - One more for the record keepers.

VOL 3 No.1 NOV '89
BASIC EXTENSION - Windows and Icons the easy way.
B-RAID - Vertical scrolling shoot'em up.
DISKONOMISER - Prudent disk block saving.
HELP - Design your own information help screens.
ORSITAL - An arcade style game with a difference.
PROGRAM COMPARE - Basic program development has never been easier.
RASTER ROUTINES - A few colourful demos.
SPRITE EDITOR 1 - A no nonsense basic sprite editor.
WABBIT - Help the rabbit collect his carrots.

VOL 3 No.3 JAN '90
4 IN A ROW - Connect a row of counters.
EROGS IN SPACE - Leap to safety across the space lanes.
BLACKJACK - Don't loose your shirt.
LORD OF DARKNESS - Defeat the evil lord in true adventure style.
MARGO - Fly around and collect jewels and fuel.
JETRACE 2000 - Have you got what it takes to be best?
ULTIMATE EONT EDITOR - Create your own screens, layouts and characters.
SELECTIVE COLOUR RESTORER - Design your own system start up colours.
6510+ UNASSEMBLER - Transform 6510+ M/C into source with labels.
TRIVIA CHALLENGE - The first of 3 files for this superb game.

VOL 3 No.4 FEB '90
COLOUR PICTURE PRINT - Download your favourite colour screens.
BASE-ED2 - An update to our popular database system.
1ST MILLION - Play the market in this strategy game.
FM-DOS - Enhance your drives operating system.
GEOS FONTS - A further 4 fonts for Geos users.
HASHING IT - Relative file programming made easy.
MULTI-SPRITE - Make full use of up to 24 sprites.
DIRECTORIES EXPLAINED - Find your way round the directory jungle.
TRIVIA CHALLENGE - The 2nd part.

VOL 3 No.5 MAR '90
PLAGUE - Become your planets Guardian and Defender.
SURROUND - Reversi on the C64
GEOS FONTS - The last of 12 new Geos fonts.
SCREEN SLIDE - Create your own slideshows.
JOYSTICK TESTER - Put your stick(s) through the mill.
COLOUR MATCHER - Mastermind for the younger players.
SCREEN MANIPULATOR - Full use of the screen now obtainable.
VIDEO RECORD PLANNER Keep tab on your home recordings.
TRIVIA CHALLENGE - The 3rd and final part of the game.

Back issues of CDU are available at £3.25 per issue, which includes postage and packing. All orders should be sent to:- Select Subscriptions Ltd, 5, River Park Estate, Berkhamsted, Herts, HP4 1HL. Please allow 28 days for delivery.

CONTENTS

Volume 3 Number 11 September 1990

ON THE DISK

IN THE MAGAZINE

Publisher: Hasnain Walji
Group Editor:: Paul Eves
Consultant Editor: Stuart Cooke
Technical Assistant: Jason Finch
Advertisement Manager: Deborah Brennan
Designer: Helen Saunders
Distribution: S.M.Distribution
Printed By: Gibbons Barford Print

Subscription Rates

UK	£33.00
Europe	£39.00
Middle East	£39.30
Far East	£41.60
Rest of World	£39.70 or $69.00

Airmail rates on request
Contact: **Select Subscriptions.** Tel: (0442) 876661

Commodore Disk User is a monthly magazine published on the 3rd Friday of every month. Alphavite Publications Limited, 20, Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF Telephone (0908) 569819 FAX (0908) 260229 For advertising ring (0908) 569819

EDITORS COMMENT

There I was sitting at the blank monitor screen wondering what to say for this months editorial spot, when I was suddenly brought back to reality by the noxious sounds of the telephone bell. To say the caller was somewhat irate would be an understatement. "What the heck is happening with my submission" he wished to know. "Don't you know its now nearly eight months since it was accepted for publication, when is it going to appear in print". The simple answer that I gave him, and this goes for EVERYONE else that is waiting to see their masterpieces appear in print is this; I cannot give ANYONE a definite date for publication. Please try to remember that we only publish approximately ten programs each issue. We also TRY not to repeat programs too often. That is to say, if we publish a SPRITE EDITOR, we try not to publish another one too soon. Therefore, trying to select original programs gets harder and harder all the time. On top of this, don't forget that we are accepting programs each and every week, therefore the library of accepted programs is growing at a greater rate than the publication rate. All I can say is this: PLEASE be patient if you have had a program/article accepted for publication. We WILL publish it eventually, it's just a matter of time. (If it is of any consolation to you all, my own programs, DISK TOOLBOX and LIBERTE took nearly 15 months before they appeared in print...and I'm the Editor..!!).

Anyway folks, once again I thank you all for your fantastic support and letters of gratitude this past month or so. It really is heartwarming to know that the hard work you put into producing a good quality (which it is) magazine, is so richly received by it's readers. Now on with the show....

DISK INSTRUCTIONS

Although we do everything possible to ensure that CDU is compatible with all C64 and C128 computers, one point we must make clear is this. The use of 'Fast Loaders', 'Cartridges' or alternative operating systems such as 'Dolphin DOS', may not guarantee that your disk will function properly. If you experience problems and you have one of the above, then we suggest you disable them and use the computer under normal, standard conditions. Getting the programs up and running should not present you with any difficulties, simply put your disk in the drive and enter the command

LOAD"MENU",8,1

Once the disk menu has loaded you will be able to start any of the programs simply be selecting the desired one from the list. It is possible for some programs to alter the computers memory so that you will not be able to LOAD programs from the menu correctly until you reset the machine. We therefore suggest that you turn your computer off and then on again, before loading each program.

HOW TO COPY CDU FILES

You are welcome to make as many of your own copies of CDU programs as you want, as long as you do not pass them on to other people, or worse, sell them for profit. For people who want to make legitimate copies, we have provided a very simple machine code file copier. To use it, simply select the item FILE COPIER from the main menu. Instructions are presented on screen.

DISK FAILURE

If for any reason the disk with your copy of CDU will not work on your system then please carefully re-read the operating instructions in the magazine. If you still experience problems then:

1. If you are a subscriber, return it to:
   Select Subscriptions Ltd
   5, River Park Estate
   Berkhamsted
   Herts
   HP4 1HL
   Telephone: 0442-876661

2. If you bought it from a newsagents,
   then return it to:
   CDU Replacements
   Interceptor Group
   Mercury House
   Calleva Park
   Aldermaston
   Berkshire RG7 4QW
   Telephone: 0734-817421

Within eight weeks of publication date disks are replaced free.

After eight weeks a replacement disk can be supplied from Interceptor Group for a service charge of £1.00. Return the faulty disk with a cheque or postal order made out to INTERCEPTOR GROUP and clearly state the issue of CDU that you require. No documentation will be supplied.

Please use appropriate packaging, cardboard stiffener at least, when returning disk. Do not send back your magazine, only the disk please.

NOTE: Do not send your disks back to the above address if its a program that does not appear to work. Only if the DISK is faulty. Program faults should be sent to: BUG FINDERS, CDU, Alphavite Publications Ltd, Unit 20, Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF. Thank you.

BANKING 128

A very simple, yet effective program for keeping a track of your bank account.

RICHARD TANNER

The following program is for C128 users with 80 column capabilities on their monitors. It will NOT work in 40 column mode. Secondly, before using the program 'Banking 128', ensure you copy the file onto a blank disk. This is because the program creates it's own files and will need lots of disk space. Last, but not least, the program on the disk is a Basic version so that you can tailor it to meet your own needs.

THE NITTY GRITTY

When the program is loaded and run for the first time, you will be told there is no data file on your disk This is correct because you have no files at present. From the displayed menu, select number 1, which creates a blank datafile. The program asks for a date, this must be valid and of the right format (DD/MM/YY). Don't worry, anything wrong will be rejected Once the datafile has been created, press any key and the program will try again to load the datafile. Now you will get another message, that there is no direct debit file. This again is correct, because you do not need to have one. This will be explained later. You now have to enter today's date, same format as before. The main menu is then displayed.

MAIN MENU DISPLAY

OPTS	EXPLANATION
1	Firstly this sorts the records, then saves them to disk.
2	Inputs credits into the system. i.e.; money into the account (the explanation field is totally free text, but you must put something in).
3	Inputs debits into the system. i.e.; money out of the account. (explanation field is as above, see option 8 for reason).
4	Displays the current state of the account (total money).
5	You can alter or delete a record, but you must have the date of the record handy.
6	Print a statement to the screen or printer starting from ?
7	Sorts the records into date order, you should hardly ever need to use this because every time you save your files, this is done for you.
8	Lists all your expenses (Debits), type in the text to search on (from the explanation fields on Credit and Debit) and all the marching expenses are displayed and totaled. This is very handy for adding up all your cheques etc., don't forget that you need to give a good explanation in 2+3.
9	Takes you to the Direct Debit Editing menu, see below.

DIRECT DEBITS

This is where you enter your direct debits ro be automatically taken out when loading the program again. You supply the starting date of the Direct Debit, the text which when added into your account records, comes out in the explanation field, and the amount.

This menu is mainly the same as the other one, but don't forget this is another file and you must save it with option 4, if you don't then when you load the program back in, all will be lost.

Once you have set up your Direct Debits, save them, and now the program is all set up for your first automatic withdrawal. Next time you load the program, all you will need to do is give today's date, this must be correct else the program will take out the direct debits missing from your files.

The Direct Debits check the month before today's date, this month and next month. If any of these files are missing it automatically adds them in. The best thing to do with the program is to experiment. If you do not have any direct debits, don't worry, the program will just skip the automatic direct debit part and put you straight into the main menu.

If you have any queries regarding this program, I will be pleased to help. Just write in to me c/o the magazine I hope that this program tidies up your bank details for you.

DISK DRIVER V4

Overcome the difficulties of talking to your drive with this latest drive utility program.

PAUL COCHRANE

The Commodore 64's Basic, as all programmers know, is sadly lacking many features. An ideal example of this is when the user wishes to communicate with the disk drive. 'Disk Driver V4' has been devised to take the workload off the programmer when they want to use the disk drive while they are creating their masterpieces.

Some of you will be saying "But I have already got these features in my extended Basic", granted. Most extensions however, provide these features by adding extra commands. Disk Driver V4 goes one better and provides all these extra utilities at the press of a Function key. The format for which is shown here:

F1 - Display directory
F2 - Validate disk
F3 - Load a file
F4 - Save a Basic file
F5 - Scratch a file
F6 - Rename a file
F7 - Read the error channel
F8 - Initialise disk

Most of the facilities offered are self explanatory, so I will describe how to respond to the computer when prompted.

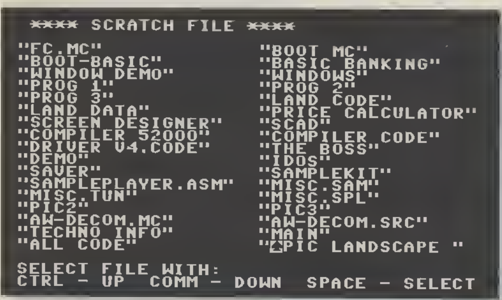
When the message 'sure (y/n)' appears, press the "Y" key if you are sure it's what you intend, otherwise press the "N" key. When a list of the files on the disk are shown, the computer expects you to select the file which you want to carry out the operation on. The highlighted cursor can be moved up through the files by pressing the 'CTRL' key, and can be moved down through the files by pressing the 'COMMODORE' key. Once the cursor is over the file that you wish to select, press the space bar to select it.

When the computer responds with 'new name' or 'save name', it expects you to type in the name that you wish to call the file. The RUN-STOP key can be used to exit this mode.

If you select the same function key twice without pressing any other key, the key will not register. This is to prevent the key being hit twice quickly which could cause the computer to crash. To re-enable the function key press any other key.

PROGRAM INFORMATION

The 'Disk Driver V4' machine code resides in the area of memory from $C900-$CF00. The program works by altering the 'KEYLOG' vector at $028F to point to the Disk Driver routine. As 'KEYLOG' is a kernal vector, it is reset upon using RUN-STOP/RESTORE. To restart the program enter SYS51456.



```
 **** SCRATCH FILE ****

"FC.MC"                    "BOOT MC"
"BOOT-BASIC"               "BASIC BANKING"
"WINDOW DEMO"              "WINDOWS"
"PROG 1"                   "PROG 2"
"PROG 3"                   "LAND CODE"
"LAND DATA"                "PRICE CALCULATOR"
"SCREEN DESIGNER"          "SCAD"
"COMPILER 52000"           "COMPILER CODE"
"DRIVER V4.CODE"           "THE BOSS"
"DEMO"                     "IDOS"
"SAVER"                    "SAMPLEKIT"
"SAMPLEPLAYER.ASM"         "MISC.SAM"
"MISC.TUN"                 "MISC.SPL"
"PIC2"                     "PIC3"
"AW-DECOM.MC"              "AW-DECOM.SRC"
"TECHNO INFO"              "MAIN"
"ALL CODE"                 "▮PIC LANDSCAPE "

SELECT FILE WITH:
CTRL - UP  COMM - DOWN  SPACE - SELECT
```

AUTOBOOT 128

C128 users can now get your CDU disks up and running with this autoboot program.

```
0B00 BYT $43,$42,$4D        ; 'cbm'
0B03 BYT $00,$00,$00        ; boot sector control codes
0B06 BYT $00,$4D,$45
0B09 BYT $4e,$55,$00        , 'menu'
0B0C BYT $00
0B0D SEI
0B0E LDX #$00               ; copy program to autostart
0B10 LDA $0B1C,X            , $8000 from $0B1C to end
0B13 STA $8000,X
0B16 DEX
0B17 BNE $0B10
0B19 JMP $FF4D              ; C12B vector - GO64
0B1C BYT $09,$B0,$09
0B1F BYT $B0,$C3,$C2
0B22 BYT $CD,$38,$30        , 'CBM80'
0B25 JSR $FDA3              ; init I/O
0B2B JSR $FD50              ; Init system constraints
0B2B JSR $FD15              ; reset KERNAL
0B2E JSR $FF5B              ; init screen timers etc
0B31 CLI
0B32 JSR $E453              , init vectors
0B35 JSR $E3BF              ; init Basic
0B3B JSR $E422              , check memory/title/NEW etc
0B3B LDA #$05               ; file number
0B3D LDX #$08               ; device
0B3F LDY #$01               , secondary address
0B41 JSR $FFBA              , kernal SETLFS
0B44 LDA #$04               , filename length
0B46 LDX #$3C               ; name address (lo)
0B48 LDY #$80               , name address (hi)
0B4A JSR $FFBD              ; kernal SETNAM
0B4D LDA #$00               ; $00=load
0B52 LDX #$FF               ; reset stack pointer
0B54 TXS
0B55 JMP $E386              , exit via error routine
```

KARL HOPE

You will all be aware that the C12B has an autoboot feature for disk users, this useful tool could be a great time saver for us C12B users that purchase CDU each month. We simply have to insert the disk and turn on the computer to access the menu.

For your convenience, there are two copies of the program on this months disk. One being the machine code (which boots to $0B00 - C12B), the other being a Basic listing of the disassembly.

For those that may be interested, here is a copy of the listing.

NOTE: Those of you who are curious and have studied the C64 ROM will notice that in a normal Initialise the C64 will reset the stack to #$FB whilst I have set it to #$FF. The reason for this is that the C64 holds return addresses after a JSR on the stack. My program was loading the files from the menu and then loading the menu again straight away. The extra four bytes usually on the stack will have been the two addresses at the beginning of the autostart program, which naturally point to my program, this being the reason why it constantly reran

READING BETWEEN THE LINES

Any adventurerer will tell you that an adventure is only as good as the parser. Now you can develop your own with relative ease.

GORDON MOYES

Have you ever told a computer to 'Chop down the western door with the axe', then had a reply like 'where did you learn English?'. The "Parser" utility presented here may come to your assistance.

This parser is intended for use as a front end for adventure programs, but can be used anywhere that you need a computer to understand commands. It was written in machine language because Basic is simply too slow. This does not, however, mean that the main program must also be in machine language.

THE SYNTAX

The first thing that must be done when writing a parser is, of course, to define the syntax of the language. The basic command structure for this parser is (verb) (verb supplement) (object1) (verb supplement) (preposition) (object2), where an object is defined as (owner) (adjective) (noun). EG: Pick up the leather saddle under the big tree. At the end of a command, there may be a full stop, comma, quote or joiner which may seperate another command.

Note that, although the verb supplement can appear in one of two places, it can only be in one place in any one sentence. You cannot say 'Pick up the axe up'! Also, an object may be replaced by a pronoun. The pronoun will be taken as the last owner/adjective/noun combination used.

If speech is involved, the beginning of a command will be (say verb) (say preposition) (object3) ("). Quotation marks are required around the speech so the parser can tell where it ends when there is more than one command in the one sentence. IE: In the sentence, Say to the tall Elf go north and get the axe; who is going to get the axe, you or the Elf? Things become much clearer if you write; Say to the tall Elf "go north and get the axe"

Many outputs are therefore required to analyse every word. Table 1 lists all of the outputs from a combination routine, which greatly reduce the number of variables that need to be used by the main program.

ACTION/OBJECT COMBINATION

The action and object combinations are both called by SYS49161. Their purpose is to reduce the number of variables that the main program must work with. The action combination combines the verb and the verb supplement into one variable, the action. It works like this: If the verb supplement is equal to zero (not present) the action equals the verb. If there is a supplement then the routine checks through the action list to see if it can match both the verb and the supplement to an action. If it does not succeed then it outputs a 255.

The object combination routine is similar to the action combination routine, except that it combines an adjective and a noun and an unlimited number of adjectives may appear in the object lists, instead of just one. The owner is not combined and is more or less irrelevant.

THE MAIN PROGRAM

Lines 7000-7070 define the variables in the machine code program. Note that FNP(X) is the same as PEEK(50040+X) throughout the program.

Lines 7080-7130 input a command string from the user and puts it into the parser input buffer along with its length.

FNP(X)	output		
0	Verb	1	Verb position
2	Verb Sup	3	Sup position
4,5,6	Owner 1,2,3	7,8,9	Owner position
10,11,12	Adj 1,2,3	13,14,15	Adj position 1,2,3
16,17,18	Noun 1,2,3	19,20,21	Noun position 1,2,3
22	Preposition	23	Preposition position
27	Say verb	28	Say verb position
29	Say prep	30	Say prep position
24	Error (see table 2)		
25	More; 0 if end of line reached. 1 if more		
26	Say More; 1 if still in quotes and more =1		
31	Action		
32	Object 1		
33	Object 2		
34	Object 3		
PA	Position in word buffer parser is up to		
BF	Start of buffer		
OU	Start of output variables		

TABLE 1 - Parser Variables

0	No error
1	There is no noun with object 1
2	There is no noun with object 2
3	There is no noun with object 3
4	Some sentence is left over after parse is done
5	No first quote in say command
6	Word is not recognised at all

.. (The position of the word that the parser got stuck at is in PA)...

TABLE 2 - Errors

Line 7140 points the input buffer pointer to the first character and resets the more and saymore variables so the parser can function properly.

Line 7150 calls the parser.

Line 7160 checks for any confusion between a verb supplement and a preposition. If there is no preposition, a second object and a supplement (which can also be a preposition i.e. on/to) then they are swapped before line 7170 calls the combination routines.

Lines 7180-7500 print the parser outputs.

Lines 7300-7320 print out the word that the parser got stuck at if there was an error. Note that unrecognised objects and actions do not show up at this stage.

Lines 7330-7410 check through the three objects to see if they are all recognised or not present. If they are unrecognised, the program prints out the unrecognised section by using the print on list routine.

Lines 7420-7490 check to see if the action is recognised. If it is not then the unrecognised action is printed out with the print on list routine.

Lines 7510-7550 wait until the space bar is pushed before printing out another list, if there was more than one command on the line that was just entered.

PRINT FROM LIST

When the inputed object is unrecognised by the main program, it would be handy to be able to tell the user this. To use this command, POKE PA with the list number (see table 3), POKE PA+1 with the words position in that list and SYS49158. The main program uses this command in lines 7370-7400 and 7450-7480 to let the user know of unrecognised objects and commands.

PRINT FROM BUFFER

This command is used to print out unrecognised words. It is called by SYS49155 when PA contains the starting position of the word in the buffer. Note that when an error occurs, PA will contain the starting position of the word that the parser got stuck at. The main program uses this command in line 7310 every time an error occurs to let the user know where it became stuck. The error number will indicate why the parser became stuck (see table 1).

THE WORD READER

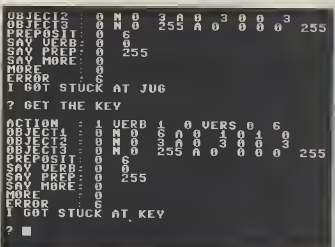
The purpose of the word reader is to read in the computers vocabulary. You can change this around as much as you like to suit your own adventure. The starting address can be changed to suit your needs, but remember to protect it from Basic. Every word stored is followed by a number. This is the word number that is returned by the parser and is used to group words that mean the same thing. The word position number that is returned distinguishes between words that mean the same thing so that they can be printed from the data lists.

Every item of action data must take the form (action) (verb) (verb supplement), with a zero byte marking the end of the list. The first entry of 1,200,1 translates to: If verb is 200 (pick) and supplement is 1 (up) then the output will be 1 (the same as get or take).

Items on the object list are stored as (object) (noun) (adj1) (adj2) . (adj n) (255), again with a zero byte marking the end of the list. Note you can have as many adjectives as you require in the list. The first entry of 5,5,5,6,255 therefore means that if the noun is 5 (horse) and the adjective is either 5 (skinny) or 6 (small/short) then the output will be a 5. Note that the same output is

0	Verbs
1	Verb Supplements
2	Nouns
3	Adjectives
4	Owners
5	Pronouns
6	Joiners
7	Prepositions
8	Say verbs

TABLE 3 - List Numbers



```
OBJECT2   :  0 N 0   3 A 0  3 0 0  3
OBJECT3   :  0 N 0  255 A 0  0 0 0   255
PREPOSIT  :  0  6
SAY VERB  :  0  0
SAY PREP  :  0  255
SAY MORE  :  0
MORE      :  0
ERROR     :  6
I GOT STUCK AT JUG

? GET THE KEY

ACTION    :  1 VERB 1  0 VERS 0  6
OBJECT1   :  0 N 0   6 A 0  1 0 1  0
OBJECT2   :  0 N 0   3 A 0  3 0 0  3
OBJECT3   :  0 N 0  255 A 0  0 0 0   255
PREPOSIT  :  0  6
SAY VERB  :  0  0
SAY PREP  :  0  255
SAY MORE  :  0
MORE      :  0
ERROR     :  6
I GOT STUCK AT KEY

? █
```

ON THE DISK

OBJECTS		
1	North	
2	South	
3	West	
4	East	
5	Horse	, small/skinny
6	Elf	; short/fat
7	Elf	; tall/blue
8	Axe	; pointy/wood
9	Armour	; rusty/metal
10	Tree	; big/wooden
11	Bars	; large/metal
12	Saddle	; leather
13	Goblin	; big/fat/orange
14	Door	; north
15	Door	; south
16	Door	, west
17	Door	; east
200	Elf	
201	Door	

ACTIONS	
1	Get/take/pick up
2	Discard/leave/put down/ throw away
3	Kick/hit/kill
4	Throw
5	Go/move/ride
6	Climb/ascend/go up/climb up
7	Descend/go down/climb down
8	Chop down
9	Dig out
10	Wear
11	Open
12	Close
13	Put
200	Pick
201	Chop
202	Dig

SAY VERBS	
1	Say/tell
2	Shout/yell

JOINERS	
1	And/then

OWNERS	
1	A/an/the/my

PRONOUNS	
1	It/them/him/her

PREPOSITIONS	
1	At
2	With/using
3	Near/around
4	Under
5	On
6	To

TABLE 4 - Parser Vocabulary

arrived at if no adjective is supplied. The main use of this is to check that only valid adjectives are used, but it can also be used to seperate between two different Elves, as in the program here.

The code presented here is only a front end for an adventure program. It is up to you to substitute the sample main program given here with your own adventure program. Many happy adventures to you!!

EDITORS NOTE:- I think that this program enhances our 'Adventure Writing' series quite considerably. You should be able to put your new gained knowledge to some good use in the very near future ..PAE!

C64 AMIGA C128

ONLY £149

C64 DISK DRIVE FREE GEOS

COMMODORE 1541 CII DISK DRIVE, SLIMLINE CASE, POWER SUPPLY UNIT, 5¹/₄" DRIVE £3 50 + P&P
THE ONLY DRIVE 100% COMPATIBLE WITH THE C64/128

NEW **AMIGA 500** ONLY £379
FLIGHT OF FANTASY
★ F29 RETALIATOR ★ RAINBOW ISLAND
★ ESCAPE FROM THE PLANET OF THE ROBOT MONSTERS
★ DELUXE PAINT II £5.00 P&P

NEW **LIGHT GUN** £36.50
C64 LIGHT GUN PLUS
GAMES AND UTILITIES
LIGHT FANTASTIC UPGRADE PACK
£3.50 P&P

PRINTERS

COMMODORE MPS1230	£159.00
COMMODORE MPS COLOUR	£219
SEIKOSHA SP 180V (C64)	£149 00

C64 LIGHT FANTASTIC
LIGHT FANTASTIC PACK INCLUDES - LIGHT GUN 3D GAMES, PAINT PACKAGE, DATA RECORDER, GAMES
£140.00 + £5 00 P&P

100% ERROR FREE **DISKS**

10x3 5" DSDD	£9 50
10x5 25" DSDD ..	£4.50
50 DISK BOX 5 25"	£5 99
100 DISK BOX 5 25"	£6 99
80 DISK BOX 3 5" . ..	£6.95

C64 DATA RECORDER	£24.50
'LOAD-IT' DATA REC	£35.00
C64 POWER SUPPLY ...	£19.95
C64 MOUSE, MOUSE MAT & HOLDER . .	£26 50
KONIX JOYSTICK	£11.50

CARRIAGE £1.50

MOUSE HOLDER	£2.50
MOUSE MAT	£4 50

C.M.S CROFTON MICRO SUPPLIES
45 WHITBREAD ROAD
BROCKLEY LONDON SE4 2BD
081-469 3246

COMPUTERS	
Amiga 500 FLIGHT of FANTASY PACK comprising TV Modulator/F29 Retaliator/Rainbow Island/ Escape from Robots/DPaint II	355 00
Amiga 500 BATMAN PACK Batman/F18/NZ Story/DPaint II	385.00
Amiga 500 CLASS OF THE 90's Education Pack	529.00
Amiga 500 + 1084S Stereo Col Mon	595 00
Commodore 64C LIGHT FANTASTIC Pack comprising 64C + C2N Light Gun + 3D Glasses + 6 Games + Paint Prog + Music Prog + Typing Tutor + Arcade Construction Kit + Audio Tape Align	139.99
Commodore 64C WORLD CUP Pack comprising 64C + C2N 2 Joysticks 5 Games	149.95
Commodore PC Starter Packs (AS SEEN ON TV)	FROM 508 35
PRINTERS	
Citizen 120D Parallel or Commodore	129 95
Star LC-10 Parallel	155 00
Star LC 10 Colour Parallel	205 00
Star LC24 10 24 pin Multi-font 170/57cps	235 00
All Okimate 20 consumables normally in stock	PHONE
MONITORS	
Commodore 1084S Stereo Colour Mon	259 00
Philips 8833 Stereo Colour Monitor	245.00
Philips 7502 Green Screen Monitor	85 00
MICELLANEOUS	
Amiga 512K RAM/Clock Exp for A500	95 95
Commodore 1541 II Disk Drive	129 00
Commodore C2N Data Recorder	29 95
External 3.5" Disk Drive for Amiga	69 95
Power Supply for C64	29 45
Super G Parallel Interface for C64/128	34 95
Star NL-10 Interface for C64/128	39 00
Surge Protector 13A Plug	12 95
Surge Protector 3-Way Adaptor / 4-Way Dist Unit	19 95/15 95
SOFTWARE	
VizaWrite 128 + Spellcheck	49 95
VizaStar 128 Spreadsheet & Database	69 95
Superbase 64 or 128	29 95
Superscript 64 or 128	32.95
Tasword 64 40/80 Col WP Tape or Disk	24 95

ALL PRICES ARE INCLUSIVE OF VAT AT 15%
CARRIAGE £5 (EXPRESS £10)
Prices subject to change without notice E&OE

Delta Pi Software Ltd

8 Ruswarp Lane, WHITBY, N Yorks YO21 1ND.
Tel 0947 600065 (9am-7pm)

IDOS

Getting to grips with Drive operations is easier with Interactive Disk Operating System.

RICHARD DAY

If you are new to a Commodore disk drive you are problably thinking "where's the Commands?" or "How do I delete a file?". Well believe it or not, the Commodore disk drives do have commands but they are VERY difficult to get used to. To delete a file from a disk you have to type in the following;

OPEN15,8,15,"S0:filename":CLOSE15

To delete a file on a BBC machine you only have to type;

*DELETE filename

Now, which is the easiest to remember? No competition, the BBC command system beats Commodores hands down. Yet the Commodore disk drives are probably the most advanced and versatile disk drive systems available for an 8-bit machine.

A BIT OF HISTORY

So why are the Commodore disk drives the most difficult to use? It all lies with a decision made by Commodore a long time ago when the VIC-20 was screaming for a disk drive. As you will probably know, the VIC-20 only had 3.5K, so there was no way that a whole disk operating system could be crammed into this micr-machine. Commodore in their infinate wisdom therefore decided to put the disk operating system into the drive itself. This proved to be both a boon and a bain, this meant that only slight alterations had to be made to the VIC's ROMs and thus saving a lot of memory, it also meant that the disk drive could be doing something while the computer did something else It also meant, (and this has only been utilised recently), that the disk drive acted like a computer in itself and two could be connected together without a computer; once they had been programmed; and talk to each other. They could for example, copy disks from one drive to the other while the computer did something totally different.

This was impossible on any other disk drive system. One problem was that the disk drive system was serial which meant that the whole system was very slow. Unfortunately, or fortunately depending on how you look at it, the VIC-20 disk drive had to be compatible with the all-new all-singing, all-dancing new machine, the Commodore 64. So the same ROM-on-drive, slow-serial system was retained. And not much has changed since then. Anyway, back to the reason for this article, to get the drive system communication through the serial cable has to be done manually, hence the complicated commands. Well, now you can say goodbye to those commands because I have devised a program which handles all those commands through a BBC-like interpreter.

THE INTERACTIVE DISK OPERATING SYSTEM (IDOS)

IDOS is an Interactive program for use with the C64 + disk drive, which allows the user to manipulate their disk with ease. Commands range from the simple LOADing and SAVing of program files to locking files against erasure. Other commands allow the disassembling of machine code programs directly off disk without first loading them into memory.

All commands are accessed through a very Basic-like Interpreter which allows abbreviations by the use of a full stop (.). This also allows ALL commands to be accessed from outside the Interpreter I.e. from Basic programs or direct mode. IDOS does not need any sort of initialising except LOADing, so once the program has be LOADed off disk it is ready for use, no SYS calls to remember!

IDOS interacts with the operating system by use of a vector system which the C64 uses to test if there is a cartridge installed. Locations $8004 to $8009 usually hold garbage, but if it contains $C3, $C2, $CD, $38, $30 ('CBM80') then the machine gives control over to the vector held in $8000 and $8001. IDOS simulates this so on pressing RUN/STOP and RESTORE the machine jumps to the address held in $8002 and $8003. Usually this would stop any warm starts but a bug (???) in the operating system also does this if only the RESTORE button is pressed, so IDOS is run by pressing RESTORE If RUN/STOP is also pressed then a normal warm start occurs, very useful for breaking out of IDOS or any other program.

THE COMMANDS

The commands are shown in BOLD; < and > denote single parameters, e.g <filename> means a filename is required NOT enclosed by quotes. Note: any names that need to contain spaces MUST use shifted spaces, (IDOS automatically changes them to normal spaces), as IDOS uses spaces as parameter separaters. Any parameters that are enclosed by [and] are optional and can be missed out. All <addr>'s are in HEXadecimal, all other numbers are in DECimal. Filenames may contain wildcards (* and ?) for selective operations. If no filename is given, '*' is used instead.

LOAD <filename> [<addr>]
This loads a program under the filename <filename> from the disk into memory at address <addr>, if no address is given then the default is the address specified by the program header (i.e. where it was saved from. Minimum abbreviation. L

Example:
>LOAD HIRES.CODE

NOTE: the LOAD/SAVE operations do not use the normal ROM routines for LOADing,SAVEing, instead IDOS uses it's own. By using the ROM routines OPEN,CHKIN,CHKOUT, CHROUT,CHRIN it does read/write operations manually. The advantage of this is that the number of blocks loaded can be shown, so you can immediately see how much of the program has loaded. It also means that IDOS has control all the time so the LOADing and SAVEing can be stopped by pressing the RUN/STOP key. Even better, it allows relocation of LOADing and SAVEing data.

VERIFY <filename> [<addr>]
This is similar to the load, so the parameters are the same. Unlike the LOAD though, the program is not actually entered into memory, just checked against the revelant memory. At the end of the VERIFY, the number of mismatched bytes is shown, unlike the ROM VERIFY routine which just shows whether all of the two memories match, this routine gives actual numbers so it is possible to see how drastic the problem is. (Usually with Machine Code programs, the program alters itself and so the version on the disk will be slightly different from the version in memory, the normal VERIFY would give VERIFY ERROR and immediately the user is convinved that the version on disk is corrupt)

Example:
>VERIFY HIRES.CODE
Minimum abbreviation VE

SAVE <filename> [L] [<addr>] [<addr>] [<addr>]
This is probably the most useful command, it allows a block of memory to be saved much like the BASIC SAVE command but a number of useful additions have been made. Firstly the command automatically uses 'Save & Replace', (Arrgh! I hear some of you cry, there has been a lot of rumours in the corridors of Commodore about the safety of this command on the 1541, but there has been just as many rumours about ways to beat this bug. One of the satest is using @0: instead of just @ Anyway, I own an Oceanic OC-118 which does not have this bug).. [Editors extra note: I have used @0: since 1984 and have NEVER had a problem of any description....Ed!!), which replaces the current version on disk with the new version. The 'Save & Replace' can be switched off by the REPLACEOFF command (see later).

The first [<addr>] is the optional start address, if it is not given or '0' is used then the start of current Basic program is used. (i.e held in locations 43 and 44).

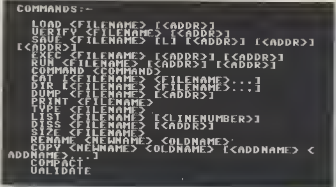
The second [<addr>] is the optional end address, if it is not given or '0' is used then the end of the current Basic program is used (i.e. held in locations 45 and 46).

The third [<addr>] is the optional relocation address, if it is not given then no relocation occurs. This allows a block of memory to be saved from one area in memory but it will load into another area of memory. Very useful when saving sprite or character data.

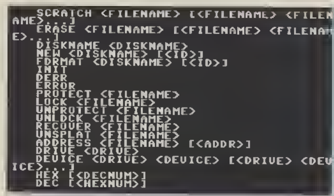
The [L] is a very quick and useful way to protect a file against accidental erasure, it locks the saved file so it cannot be erased (unless you unlock it of course).

Examples:
SAVE IDOS L 8000 A000
Will save IDOS onto disk and lock it (From $8000 to $A000 where IDOS exists).
SAVE PROGRAM
Will save a Basic program onto disk, but does not lock it.
Minimum abbreviation: S

EXEC <filename> [<addr>] [<addr>], RUN <filename> [<addr>] [<addr>]
These commands load and execute a program off the disk. It uses the same parameters as the LOAD. The first <addr> is the load address and the second is the execution address. If no address is stipulated the program is



```
COMMANDS:-
   LOAD <FILENAME> [<ADDR>]
   VERIFY <FILENAME> [<ADDR>]
   SAVE <FILENAME> [L] [<ADDR>] [<ADDR>]
[<ADDR>]
   EXEC <FILENAME> [<ADDR>] [<ADDR>]
   RUN <FILENAME> [<ADDR>] [<ADDR>]
   COMMAND <COMMAND>
   CAT [<FILENAME> <FILENAME>...]
   DIR [<FILENAME> <FILENAME>...]
   DUMP <FILENAME> [<ADDR>]
   PRINT <FILENAME>
   TYPE <FILENAME>
   LIST <FILENAME> [<LINENUMBER>]
   DISS <FILENAME> [<ADDR>]
   SIZE <FILENAME>
   RENAME <NEWNAME> <OLDNAME>
   COPY <NEWNAME> <OLDNAME> [<ADDNAME> <
ADDNAME>...]
   COMPACT
   VALIDATE
```



```
   SCRATCH <FILENAME> [<FILENAME> <FILEN
AME>...]
   ERASE <FILENAME> [<FILENAME> <FILENAM
E>...]
   DISKNAME <DISKNAME>
   NEW <DISKNAME> [<ID>]
   FORMAT <DISKNAME> [<ID>]
   INIT
   DERR
   ERROR
   PROTECT <FILENAME>
   LOCK <FILENAME>
   UNPROTECT <FILENAME>
   UNLOCK <FILENAME>
   RECOVER <FILENAME>
   UNSPLAT <FILENAME>
   ADDRESS <FILENAME> [<ADDR>]
   DRIVE <DRIVE>
   DEVICE <DRIVE> <DEVICE> [<DRIVE> <DEV
ICE>...]
   HEX [<DECNUM>]
   DEC [<HEXNUM>]
```

assumed to be a Basic program and is therefore RUN

Example:
>EXEC MC-GAME C000
Minimum abbreviations E for EXEC, R for RUN

COMMAND <command>
This outputs a normal DOS-type command such as S0:filename
Minimum abbreviation: C

CAT {<filename>[{<filename>...}, DIR {<filename>[{<filename>...[
These commands display the Directory of the current disk, the filenames are optional and allow for selective cataloging.

Example:
>CAT
Minimum abbreviation: CA for CAT, D for DIR

PRINT <filename>, DUMP <filename> {<addr>}, TYPE <filename>, LIST <filename> {<linenumber>}, DISS <filename> {<addr>}
All these commands display file data from the disk to the screen but in different ways. DUMP displays a Hex and ASCII dump of a file from <addr> onwards. If no address is given then the start of the file is used. PRINT prints out a file onto the screen, it does not print any control characters at all. TYPE is like PRINT but it prints out all characters including control characters. LIST lists a Basic program out from the disk to the screen from <linenumber> onwards, if no linenumber is given then it is listed from the start. DISS disassembles a machine code program from the disk onto the screen from <addr> onwards. If no address if given then the start address is used. All these commands can be used on PRG, USR and SEQ files. The output of a file to the screen can be paused by pressing CTRL, SHIFT, SHIFT LOCK or the COMMODORE KEY, or stopped by pressing the RUN/STOP key.

Example:
>DISS HIRES-CODE
Minimum abbreviations· DU for DUMP, LI for LIST, DI for DISS, T for TYPE, P for PRINT

SIZE <filename>
This shows the size of a file in blocks and bytes, It also shows where the program would be situated in memory.

Example:
>SIZE HIRES-CODE
Minimum abbreviation: SI

RENAME <newname> <oldname>
This renames a file on disk from <oldname> to <newname>

Example:
>RENAME HIRESCODE HIRES-CODE
Minimum abbreviation: RE

COPY <newname> <oldname> [<addname> <addname>...[
This copies <oldname> to <newname> on the disk, <addname>'s optional and are appended onto <newname>

Example:
>COPY HIRES-CODE1 HIRES-CODE
Minimum abbreviation: CO

COMPACT, VALIDATE
These commands validate the current disk. Equivalent to V0.

Example:
>COMPACT
Minimum abbreviations: COM, VA

SCRATCH <filename> {<filename...>}, DELETE <filename> {<filename>...}, ERASE <filename> {<filename>...}
These commands remove <filename>'s from the disk.

Example:
>DELETE HIRES-CODE1
Minimum abbreviations: SC for SCRATCH, DE for DELETE, ER for ERASE

DISKNAME <diskname>
Renames the current disk to <diskname>

Example:
>DISKNAME BASIC PROGRAMS
Minimum abbreviation: DISK

NEW <diskname> {<id>}, FORMAT <diskname> {<id>}
These commands format the current disk with the name <diskname> and ID <id>. WARNING: There is no 'Are your sure' routine, once the RETURN key is pressed the FORMatting will commence.

Example:
FORMAT NEWDISK ID
Minimum abbreviations: N for NEW, F for FORMAT

INIT
This initialises the current disk.

Example:
>INIT
Minimum abbreviation: I

DERR, ERROR
These commands read the disk error channel and displays it.

Example:
>DERR
Minimum abbreviations: DER for DERR, ERR for ERROR

PROTECT <filename>, LOCK <filename>
These two commands lock the given file against erasure. Equivalent to the L in the SAVE command.

Example:
>LOCK HIRES-CODE
Minimum abbreviations: PRO for PROTECT, LOC for LOCK

UNPROTECT <filename>, UNLOCK <filename>
These commands are the reverse of the above, i.e. they allow for the deletion of the given <filename>.

Example:
UNLOCK HIRES-CODE
Minimum abbreviations: UNL for UNLOCK, U for UNPROTECT

NOTE: A locked file has a '<' after the file type in the directory.
E.G. 33 "IDOS"   PRG<

RECOVER <filename>, UNSPLAT <filename>
These commands restore a file which has not been properly closed (i.e. has an '*' before the file type). The file can now be opened and read from so restoring it.

Example:
>RECOVER FILE
Minimum abbreviations: REC for RECOVER, UNS for UNSPLAT

ADDRESS <filename> {<addr>}
This command allows you to change the LOAD address of any file to <addr>, if no address is given, then the start of Basic is used. This is equivalent to the third <addr> in the SAVE command.

Example:
>ADDRESS SPRITEDATA 4000
Minimum abbreviation: A

DRIVE <drive>
This changes the drive you are working on to <drive>. <drive> must be between 0 and 3 or the program will respond with ERROR. ILLEGAL DRIVE. It does not alter the number of drive but the device number which the program uses (see below).

Example:
>DRIVE 0
Minimum abbreviation DR

DEVICE <drive> <device> {<drive>} {<device>...}
This alters the device number which the program uses for <drive> to <device>. The program uses 4 preset device numbers, one for each of the drives, this command alters these presets. On loading they are set up as follows:

DRIVE	DEVICE
0	8
1	9
2	10
3	11

The command alters the preset device number for drive <drive> to device <device>. This may not seem like a very useful command, but it can be. Imagine that you own all 4 drives. One of which is a 1541, two are 1581's and the last is an Oceanic OC-118N. The 1541 must have a device number of 8 because you cannot change it without opening up the disk drive, the other three do have device changers and so they are set to 9,10 (the two 1581's) and 11 the Oceanic. It is obvious that you would want one of the two 1581's to be drive 0 because they have a bigger capacity, the OC-118N would be drive 1, the other 1581 drive 2 and the poor old 1541 would be drive 3 (because of it's slowness and bugs) Normally this setup would be impossible, but with IDOS it is possible.

Example:
>DEVICE 0 8 1 9
Minimum abbreviation: DEV

HEX {<dec number>}
This simple command converts a dec number into a hex number and displays it.

Example:
>HEX 57344
Minimum abbreviation: H

DEC {<hex number>}
This is the opposite of the above. i.e. it changes a hex number into a decimal number.

Example:
>DEC F1CA
Minimum abbreviation: DEC

CALL {<addr>}
This command is the same as the Basic SYS command, it jumps to a machine code subroutine.

Example:
>CALL C000
Minimum abbreviation: CAL

RESET {<addr>}
This does a system reset and sets the Basic start address to <addr> (see below). Equivalent to SYS64738.

Example:
>RESET
Minimum abbreviation: RES

BASIC {<addr>}
This command sets the start of Basic to <addr>, if no <addr> is given then the current Basic program will be NEWed.

Example:
>BASIC 0801
Minimum abbreviation: B

COLOUR <border col> <screen col> <text col>
This command sets the colours of the screen and the colour presets of IDOS.

Example:
>COLOUR 6 15 6
Minimum abbreviation: COL

PROMPT <prompt-text>
This command sets the prompt for

IDOS, normally it is set to a '>' but this command allows you to change it. Using a backarrow inserts a carriage return in the string (CHR$(13)), the text can be anything up to 40 characters in length. To get no prompt just type in PROMP and press RETURN.

Example:
>PROMP ENTER COMMAND>
Minimum abbreviation: PROM

ERRON, ERROF, TEXTON, TEXTOFF, READON, READOFF, REPLACEON, REPLACEOFF
This set of commands are flag setters. ERRON and ERROFF control DOS error messages output. When switched off, no 00 OK 00 00's are outputted, but all others are (eg 30 SYNTAX ERROR 00 00).

TEXTON and TEXTOFF control the output of certain types of text, these are the LOADing and SAVEing messages (BLOCKS $— and ADDRESS $—— - $——) and ALL DOS error messages.

READON and READOFF control the reading of the first two characters in a file, In program files this refers to the LOAD address of the file. It is most useful in DUMPing a file as it allows you to see how long the file is (the first two numbers in the dump will be the LOAD address).

REPLACEON and REPLACEOEF control the 'Save and Replace' command in the SAVE. When it is ON then the form @0:filename will be used, when it is OFF then the form 0:filename will be used. I included this because some people are worried about the safety of the replace command.

Example:
>ERROFF
Minimum abbreviations: ERRON for ERRON, ERROF for ERROFF, TE for TEXTON, TEXTOF for TEXTOFF, REA for READON, READOF for READOEE, REP for REPLACEON, REPLACEOE for REPLACEOEE.

EXIT, QUIT
These commands exit you from IDOS to the READY prompt in Basic. This routine will break you out of the current program (whether it is Basic or Machine Code)

Example:
>QUIT
Minimum abbreviation: EXI for EXIT, Q for QUIT

HELP
This command lists all the commands with their syntax, useful if you have forgotten how to use a command.

Example:
>HELP
Minmum abbreviation: HE

KILL
This command disables IDOS and exits you from it. It is possible to restore IDOS easily (see later).

Example:
>KILL
Minimum abbreviation: K

IDOS IN USE

The program is accessed by pressing the RESTORE button only (either in program or direct mode) or by typing SYS32780. The screen colours will change to the preset colours (altered by the COLOUR command), there should be a greeting message and the disk error channel displayed (unless ERROFF or TEXTOFF has been executed).

```
INTERACTIVE DISK OPERATING
SYSTEM MK4 0
BY RICHARD DAY © 1989

00, OK,00,00
>
```

A prompt (usually '>', but altered by the PROMPT command) and a flashing cursor should appear. This means that IDOS is waiting for an input, so type in a command along with any parameters and press RETURN.

If IDOS responds with;
ERROR : COMMAND NOT RECOGNISED

Then you have not typed in a known command. To exit IDOS type EXIT, QUIT or their abbreviations. To perform a normal warm start hold down the RUN/STOP and press the RESTORE button.

ACCESSING COMMANDS FROM BASIC

I mentioned before that all the commands are available from Basic, well they are. To access any command from Basic immediate mode or program mode type;
SYS32777,<command>.
The command MUST be in a string (i.e. inside quotes), this allows the following:-
SYS32777,"DELETE "+E1$+" "+F2$
Some useful SYS calls are;

SYS32777,<command>	Execute command
SYS32780	Enter IDOS
SYS32783	Enable IDOS
SYS32786	Disable IDOS (same as KILL command)

GENERAL POINTS

IDOS resides in the block of memory set aside for cartridges ($8000-$A000), so it will not work with any sort of cartridge system installed. IDOS cannot be relocated at all, even if you were to change all the references in the program, the RESTORE entry system would not work.
Now I have a slight confession to make, there is a bug in the program, but before you start worrying it can be easily over-written. Sometimes the UNLOCK/UNPROTECT routine does not work. I do not know why, maybe my drive is playing up, but it is not permanent. If you do find this bug appearing continue using the UNLOCK/UNPROTECT command and CATaloging until the file is unlocked, believe me it will.

AND FINALLY

And finally, my advice to you is to experiment, be careful with the disk you use to begin with just incase you accidentally erase something you want. The are only a few commands which actually alter the disk, most just read off it. I have a copy of this program on every one of my disks, not because I wrote the program, but because I find it indespensible and couldn't do without it.

ON THE DISK

PRICE CALCULATOR

Calculate the buying power of any given sum of money for any year this century with this handy utility.

IAN DALZIEL

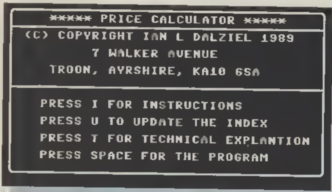
Have you ever wondered if your wages, pocket money or pension have really kept up with the cost of living? If you employ someone to do your gardening, or housework, are you sure you're keeping up their salary in real terms? Are recent increases in your annual subscriptions to magazines, clubs etc really needed to keep up with prices?

Certainly, everyone knows the current yearly increase in the Retail Price Index (RPI), but calculating prices over a longer period is another matter, even if you know just where to get the figures. Have you any idea at all how much prices have increased in the last 10 years? Well, wonder no more, for **"Price Calculator"** will answer all your questions about price increases and the 'real value' of money, not only for the last 10 years but for any period this century.

Perhaps you are a club treasurer and are preposing an increase in subscriptions, it will greatly increase your chances at the AGM if you can prove the increase is actually less than prices generally over the period. When someone tells you how cheap an item was in say 1960, you will be able to see at a glance if it was actually cheaper in real terms.

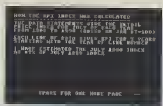
If for example, you have a son or daughter that has just started work at a salary of 5,000 pounds per year, they will be able to compare it with the salary of their parents when they started work in 1965 but at 1990 prices. You will be able to astound your friends with your newfound knowledge, and if you are a politician you will be able to prove anything with statistics!

The program is very user friendly, all you need to do is enter the price you know and the year applicable, then simply enter the year you wish to know the price of, for any year this century. The information used to perform the calculations was carefully researched over a long period of time, but anyone is welcome to check the calculations and the program has help screens to enable you to do so. You may also update the program as prices become known for future years and the program shows you how to do it. There are four worked examples in the help screens which will explain exactly what it is about.



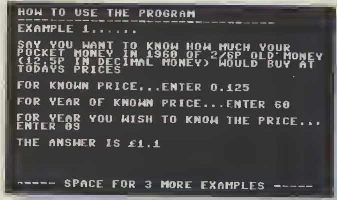
```
  ***** PRICE CALCULATOR *****
 (C) COPYRIGHT IAN L DALZIEL 1989
        7 WALKER AVENUE
    TROON, AYRSHIRE, KA10 6SA

   PRESS I FOR INSTRUCTIONS
   PRESS U TO UPDATE THE INDEX
   PRESS T FOR TECHNICAL EXPLANTION
   PRESS SPACE FOR THE PROGRAM
```



```
HOW THE RPI INDEX WAS CALCULATED
THE DATA STATEMENTS GIVE THE RETAIL
PRICE INDEX FOR JULY OF EACH YEAR
FROM 1900 TO 1990 (BASED ON JAN 87=100)
EACH LINE OF DATA GIVES RPI FOR 7 YEARS
STARTING WITH THE YEAR OF LINE NUMBER
I HAVE ESTIMATED THE JULY 1990 INDEX
AS [illegible] OF JULY 1989 INDEX

      SPACE FOR ONE MORE PAGE   —
```



```
HOW TO USE THE PROGRAM
---------------------------------
EXAMPLE 1......
SAY YOU WANT TO KNOW HOW MUCH YOUR
POCKET MONEY IN 1960 OF 2/6P OLD MONEY
(12.5P IN DECIMAL MONEY) WOULD BUY AT
TODAYS PRICES
FOR KNOWN PRICE...ENTER 0.125
FOR YEAR OF KNOWN PRICE...ENTER 60
FOR YEAR YOU WISH TO KNOW THE PRICE...
ENTER 89
THE ANSWER IS £1.1

----- SPACE FOR 3 MORE EXAMPLES -----
```

B.O.S.S.

MARC BANGS brings to you 23 new commands with his Basic Operating System Supplement. Developed with the C128's own Assembler

After reading Burghard-Henry Lehmann's articles on Extending Basic, published in Your Commodore, I decided to write my own for you. These routines were developed using the built in Assembler on my C128. Because of the way this extension has been coded, it is a simple job to extend it with your own commands. Simply add your own commands at the end of the code and make the necessary branches/jumps.

B.O.S.S. COMMANDS

CLS - Clears the screen.

FAST - Puts the 64 in 2MHz mode for a speed increase of over 5%.

SLOW - Returns to normal speed (re-enabling the screen).

OLD - Recovers a newed program.

OFF - Turns off the Basic extension.

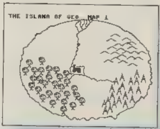
DISK - Defaults all I/O to disk. No secondary address may be passed whilst in operation.

TAPE - As above, but defaults to tape.

NORM - Returns to standard. Primary and Secondary addresses can be passed.

HELP - The all important list of commands.

CLKON - Turns on the Basic clock. The clock shows the current value of TI$ in the top right hand corner of the screen. The clock runs off the Basic interpreter, only being updated while Basic is being processed. This results in the clock stopping during an INPUT statement, or when no command is being processed. The clock will also stop if the command OFF is given. The advantage of this is that the interrupt is left unused, and can run some other background program.



CLKOFF - Turns the clock off.

HIRES - Turns on hi-resolution mode

LORES - Cancels hires mode

FILL - This changes the entire colour memory of the picture. The syntax is FILL x where x is a number in the range 0 to 255. The high nybble of this number is the paint colour, the low nybble the canvas. EG: FILL1 will give Black paint on a White canvas. You should always call FILL after using the HIRES command for the first time, otherwise pictures tend to appear scrambled.
(The last three commands came out of my use of DOODLE, but should be compatible with similar art packages).

INK x - Changes the character colour to value of x. The range is 0 to 15.

BORDER x - As above but for the border colour.

BACK x - As above but for the background colour.

AT x,y - Changes the position of the cursor to x for row and y for column. EG: AT 12,15:PRINT"HI THERE" (Note the colon before the Print statement).

ROM - Simply shows the Kernal release number of your machine

POINTER on/off - Turns sprite 0 either on or off. An interrupt is set up, which enables the sprite to be moved with a joystick in Port 2. The sprite can be moved to any position on the screen but will not wrap perfectly. The sprite definition is controlled by location 2040 decimal. I have not included a default definition, this I will leave up to your own imagination.

PCOL x - Changes the colour of the pointer. (Note that the pointer is not set as multi-colour).

POS x,y - Enables you to set the pointer position from 0 to 344 in the x axis and 0 to 255 in the y. Any value for x above 255 is handled by the program, and the x coordinate bit is set or cleared automatically. The pointer is still under joystick control, and that this use of POS will not interfere with the normal Basic use.

DEFAULT - This acts as if the following sequence had be entered. LORES:INK14:BORDER14:BACK6

If none of the above commands are recognised, an error is returned by the B.O.S.S. and program execution, if any, is stopped. This is, in effect a syntax error with a different message. The 'SPA' in the message is an abbreviation for SPANNER, which just happens to be my computer handle.

B.O.S.S. is entirely self contained in the present range of C000 to C700, which is 40152 to 50944 decimal.

SCREEN DESIGNER

Creating your own impressive screens gets another boost from this handy utility program.

A WARRINER

The utility is made up a of a couple of routines which will now be described

SCREEN COMPILER (address 52000-52775)

This program copies the current lo-res screen being displayed and creates a stand alone machine code program which will re-create the screen. The program automatically detects the video bank, screen address, character set pointer, colours and colour mode being used and restores these values when the screen is re-created.

The compiled code uses ten zero page addresses from 165 to 174 inclusive. To compile a screen, print the screen, then sys 52000,AD where AD is the address at which the code is to be compiled, the program will return the end address of the code To re-create the screen simply enter sys AD. The screen data is not just stored as a block of data, but is compressed according to the following method. The program searches from the start of the screen memory until the first character which is not a SPACE (#32) is found, this is the start of the screen. It then searches from the end of the screen, backwards, in the same way and marks this as the end of the screen. The screen data is then read in, compressed and stored When the screen is re-created the screen is cleared and the video parameters are set,the data is then read and interpreted as follows:-

If the data value is less than 128 then it is OR'ed with a variable (REVFLAG), which determines if reverse video is on or off, and stored to the screen, the current colour variable (CURCOL) is stored in the corresponding colour memory.

If the data value is greater than 127 then it is deciphered thus:-

If bit 4 is set then bits 0 to 3 hold the new colour value (CURCOL)

If bit 5 is set then REVFLAG is flipped, i.e. reverse on becomes reverse off and vice-versa

If bit 6 is set then there are a number of characters of the same value to be repeated, in which case, the next data byte represents the number of characters to be done, and, the byte after that holds the character value to be used.

More than one bit may be set at a time

If bits 0 - 6 are zero then this is the last byte and the routine exits

The program "compiler 52000" on the disk is a basic loader for "compiler code", which is the program itself

SCREEN OESIGNER/COMPILER (address 16384-28374)

This program makes it easier to design and compile screens.

When the program starts you are presented with the main menu from which you may select an option by, moving up or down the menu with the cursor keys, or, by typing the initial letter of the option, the options are as follows:-

DESIGN SCREEN

In the designer you may type in characters and change colours in the normal way, but in addition the function keys will give the following effects.

F1	-	Centre text on cursor line.
F2	-	Clear cursor line.
LOGO+F1	-	Undo last clear screen or function key.
CTRL+F1	-	Enter block mode.

In block mode you must follow the following sequence.

Move cursor to upper left corner of block and press RETURN.

Move cursor to lower right corner of block and press RETURN.

The block is now defined and may be moved around the screen with the cursor keys, in addition:-

RETURN - will print the block.

Any COLOUR key will fill the block with that colour

F7	-	reverses all of the characters in the block.
RUNSTOP	-	will print the block and exit.
CLR	-	clears the block and exits.

Continuing function key definitions

F3	-	Insert a blank line and scroll screen down.
F4	-	Scroll screen up to cursor.
CTRL+F3	-	Insert a blank line and scroll screen up.
LOGO+F3	-	Scroll screen down to cursor
F5	-	Reverse on.
F6	-	Increment screen colour.
CTRL+F5	-	Increment border colour. (SHIFT+CTRL+F5 increments both screen and border)
LOGO+F5	-	Switch cursor off, any key cursor on
F7	-	Reverse off.
F8	-	Switch Forcecolour/ Copycolour mode.

Forcecolour is the normal typing mode in which the character printed is the colour of the cursor. In copycolour mode the character is the colour of the existing character on the screen, unless, that colour is the background colour in which case the last, selected, colour is used.

CTRL+F7 - Paint with cursor colour.

In this mode as the cursor moves over the screen it colours the characters it moves over, any key, other than the cursor keys, exits this mode

LOGO+F7 - Draw with current character.

In this mode the character which is under the cursor when the mode is selected is printed as the cursor moves over the screen, the colour is determined by Forcecolour/ Copycolour.

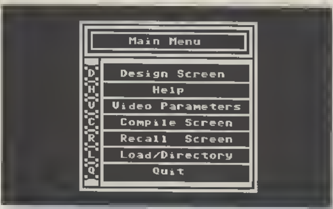
To exit the designer press RUNSTOP to return to the main menu, or, SHIFT+RUNSTOP to call up HELP screens.

HELP

This will call up the designer help screens. RUNSTOP returns to main menu any other key calls second help screen.

VIDEO PARAMETERS

On this screen you may select the video bank, screen address, character set pointer, colour mode and colours used by the designer and compiled code. You may also view the selected character set. Certain addresses are not available as they are used by the program. RUNSTOP to return to main menu.

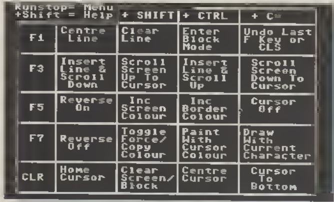


COMPILE SCREEN

On this screen you will be prompted to enter the address at which you want the designer screen code to be compiled, if you have previously compiled a screen you will be informed of the next available address to allow you to create consecutive blocks of code, pressing RETURN without entering anything will automatically compile at this address. You will not be allowed to compile code which overwrites the program, this is not a problem as you can compile the code to a different address and then use the seperate screen compiler to move it to your desired location. Once code is compiled you have the option to save the code in which case you will be prompted for a filename. Code may be compiled and recalled from under the I/O area and under the Roms, but, only code under the Basic Rom (40960-49151) may be saved. The program uses the area under the Kernal Rom (57344-65535) as a workspace and so any code compiled there may be corrupted.

RECALL SCREEN

This allows you to put a previously compiled screen into the designer, also, all of the video parameters are read from this code. If no code is found at the address you specify then you will be asked if you wish to search for code, if you do then the program will search from that address until valid compiler code is found which is then loaded into the designer, you may then exit or continue to search



Runstop= Menu +Shift = Help		+ SHIFT	+ CTRL	+ C=
F1	Centre Line	Clear Line	Enter Block Mode	Undo Last F Key or CLS
F3	Insert Line & Scroll Down	Scroll Screen Up To Cursor	Insert Line & Scroll Up	Scroll Screen Down To Cursor
F5	Reverse On	Inc Screen Colour	Inc Border Colour	Cursor Off
F7	Reverse Off	Toggle Force/ Copy Colour	Paint With Cursor Colour	Draw With Current Character
CLR	Home Cursor	Clear Screen/ Block	Centre Cursor	Cursor To Bottom

LOAD/DIRECTORY

Selecting this option allows you to view the disk directory and load from it. You are not allowed to load code whose start address is in the program memory space. When code is loaded the load start and end addresses are displayed, and, if the code is a compiler screen it is loaded into the designer and a red asterisk (*) is displayed to inform you this has happened.

QUIT

Reset computer. Re-enter program with sys 16384.

The program "Screen Designer" (2049-3014) on the disk is a basic loader and loading screen for the main program "scad".

LANDSCAPE ROUTINE

A guide to scrolling landscapes on the Commodore 64 for beginners.

EDMUND DUMBILL

First and foremost let me just point out one thing. On some older C64's, the colour memory may need to be set each time the routine is called, by POKEing to it. (55296-56295). On the more recent models, simply selecting the colour by POKEing 646,n will do the trick. Note that this is only if the routine does not at first appear to work.

One of the first questions budding young programmers ask is; 'How do I get to screen to scroll?' How indeed? First, the scrolling must be done from machine code as Basic is too slow to avoid screen flicker. Scrolling uses special hardware provided by the VIC chip. In essence, all that is needed is to update a 3 bit (0-7) counter, and let the VIC chip do the rest. This of course, does not move the characters on the screen into different screen memory locations, but it shifts the whole of the screen several pixels to the left/right up/down. There are in fact 2 scroll counters, x-direction (horizontal) and y-direction (vertical). To gain the left effect as opposed to the right effect, you must decrement the 3 bit counter, and vice versa to go right. The process is the same for the y-direction. To move characters on/off the screen, a simple piece of machine code is needed, which copies every character on the screen either one character to the left/right up/down. So now we know basically how scrolling is done, you may be asking the question; where is all the mythical registers? A quick consultation of the users guide will assist you. The y-direction is bits 0-2 of location 53265 and the x-direction is bits 0-2 of location 53270. To achieve neat scrolling you must 'shrink' the screen so you can hide the characters you are putting onto the screen in the border. If you don't do this the whole effect of scrolling is ruined as large gaps move back and forth at the side of the screen. To shrink the screen in the x-direction, you must set bit 3 of 53270 and the y-direction is bit 3 of 53265. One of the most common uses of scrolling is in providing a landscape for a spaceship etc in a game. This is most commonly done from right to left, that is decrementing the 3 bit counter 53270. I will now give a program breakdown of a landscape scrolling routine.

"LEFT A BIT, RIGHT A BIT, DOWN A BIT . . . SCROLL"

THE ROUTINE

The routine is a very simple one, providing a landscape 256 characters long by 11 lines deep. The landscape is actually interrupt driven, so your program can be getting on with its own business while the landscape scrolls along. Another aspect of interrupts is also brought in; the split screen. As the landscape is only 11 lines deep, we want the other 14 lines to stay still, so the screen is split at the 12th line. As a novelty, I have included a variable speed of scrolling. This is achieved by instead of scrolling the screen 1 pixel every 50th of a second, scrolling the screen by any amount stored in a memory location. The number in this location consequently becomes the speed number. As a further novelty, the screen is made to be in multicolour mode, and the multicolour changed constantly. So if the colour memory contains colours from 8-15 the characters with that colour assignment will appear to glitter as the multicolour changes. The routine does not actually update the colour memory, so it is advaisable only to use one colour on the screen. Before you try to use the landscape routine, you should POKF the colour memory to its proper settings for your landscape. (See note at start of article).

HOW THE ROUTINE WORKS

I shall not go into detail as to how the interrupt routine works, as it does not pertain particulary to scrolling (Eric Doyle wrote a series of interrupt articles in your commodore in 1988-1989). Every 50th of a second, the scroll routine is called. On being called, it decrements the speed counter and then scrolls one pixel to the left. The state of the scroll 3 bit counter at 53270 is actually stored separately in $C0FF. It is checked if the scroll counter has reached zero, If so, it is time to shift the screen one character to the left, and to reset the scroll counter. If you did this in Basic, you would be able to see the apparent spectacle of the whole screen shifting back 8 pixels, and then shifting back 8 pixels again with the new screen data. Obviously, machine code is too fast for the eye to see what is going on. The process of updating the 3 bit counter is continued for as long as the speed counter says. The speed counter is located at $C0FD. Shifting the whole screen one character to the left is a laborious job, of which there are two ways of going about it. The first way is the way I've used, which is to shift one row at a time with seperate commands and then to shift the new data in from $C300 onwards. This is rather a long winded way, and a better way, but more complicated, would be to use extensive indirect addressing to shift all data in and out. The advantage of a long winded way is that it is often the faster way, as in this case. There is a pointer which keeps track of where the computer is along the landscape. This counter is located at $C0FE, and is decremented every time the screen is shifted one character to the left. A few pointers are used from $C200-$C206. You can try altering them if you want, $C201-$C202 control border colour around and below the landscape, and $C203-$C204 the screen colour. $C205-$C206 control the position of the screen split. You can try splitting the screen at different places to alter the scroll, but remember that the shift routine does not take account of screen split position.

TO USE THE ROUTINE

The landscape data used by the computer is at $C300 and ends at $CDFF. The data is stored as screen codes not as ASCII data. There is no easy way to POKE in your landscape data, apart from to write a landscape designer program. It is probably best to use a character set that you have defined especially for the landscape, but I have designed a landscape that uses the default character set. To start the landscape going simply select it from the CDU menu or alternatively load it independantly and type SYS49152. For the more inquisitive amongst you the following is a breakdown of the main program code.

"TRY SPLITTING THE SCREEN AT DIFFERENT PLACES FOR GOOD EFFECTS"

$C000-$C025 - Setup variables and interrupt pointers
$C026-$C065 - Main interrupt handling routine. Deals with split screen and border colours.
$C070-$C07F - Decides whether the landscape routine should be called, and calls it if needed.
$C0BA-$C15F - Shift screen one character to the left routine. Interposed with variables as already mentioned.
$C160-$C187 - Main scroll routine.

I hope this short introduction into the art of scrolling will prove to be of some benefit to you. Especially to the novices out there in C64 land.

WINDOWS 128

Storing and recalling windows is simplicity itself with this handy C128 routine.

JASON DOIG

While designing a short program for my C128, I decided to use some windows to make the display more attractive. Then I remembered how Commodore had 'forgotten' to include any window saving routines in the Basic. Thus, I resolved to write this wedge which would record the screen contents, (including colour), everytime a WINDOW command was reached. Also wedged was an extension to the command which would close the current window and recall the old screen. Up to 9 windows can be opened (10 including the original screen).

So, how do you use the routine? First of all you will need to load in the wedge by BLOAD"WINDOWS" <return>. That will load the routines into memory from $1400 to $17FF (5120-6143). Now you should transfer the routines into both RAM banks (To avoid crashes when using a variable) and install the wedge. Enter this; SYS5381:CLR This will also reset the windows stored to zero. So any windows previously opened will be lost. It is a good idea to place this at the begining of your programs. The reason for the CLR is to clear the variable area which has been moved to make room for the windows which will be stored in BANK 1 underneath them. Now WINDOWS will be installed until the machine is reset. (after a reset, using the above SYS call should re-install WINDOWS without re-loading).

The WINDOW command operates as normal, except only 9 may be used in a row. Any more and memory is filled up resulting in a syntax error being generated (as opposed to variables being corrupted).

To restore screen contents after a window is finished with enter the command - WINDOW* - this will remove the last window opened and decrease the window counter.

There is another useful feature of this program. After opening a window it can be bordered and shadowed. Use the command SYS5696,<border colour>,<shadow colour>. This will place a border around the current window and shadow it. The colours are from 0 to 15.

One small point to note is that your variable space is reduced to about 40K. This should not cause any problems, but be careful not to go overboard with enormous arrays. The program also uses zero page 250-255 ($FA - $FF) addresses. To see the wedge in action LOAD and RUN the program 'Window demo' after BLOADING WINDOWS. One last point worth mentioning ALWAYS home the cursor after opening a new window, otherwise strange things have been known to occur (but very rarely).

[illegible]

SAMPLE KIT 64

To compliment last months program 'Sequencer 64' by Steve Carrie, I present a sample program of my own creation

IAN GOFFE

The SAMPLE KIT 64 was originally designed for use by owners of the excellent digital sound sampler from Datel Electronics. However, even those without the sampler hardware, can have hours of fun by using this kit, with the accompanying demonstration sound sample, and in the process give an indication of what sampling effects are possible on the C64.

THE KIT IN USE

The kit enables a sound sample to be 'chopped' into individual parts, known as the SAMPLE SPLITS; these 'sounds' can then be linked together in a sequence, known throughout as the TUNE. Edited tunes can be saved to disk, and data created can be used to playback the tunes within your own programs. To aid in this I have included a source file named SAMPLEPLAYER.ASM, which can be assembled by the MIKRO assembler, and CDU's very own 6510+.

USING YOUR OWN SAMPLES

The Sample Kit utilises a standard sample file, which is created with Datel Electronics "Sampler 64" software/hardware package. The kit requires you to save the sample data from Datel's utility as a full 32K file. This is a save range of 0-B, (a sample split of 1); see the sound sampler instructions manual for more details of setting the sample save range. Other sound sampler data files may be compatible with the kit, but I have only had chance to use Datel's unit. So, if you have another sampler, why not try it out!

By using a full sample range, a combination of smaller samples can be used to include a variety of instruments, thus giving more reality to the final tune.

SAMPLE KIT EDITOR SCREEN

The screen display is split into two main areas. The top of the screen comprises of the SPLIT METER, this is the display on which the sample splits are tested, and created. The area below is the CONTROL PANEL, coming complete with a selection of 'function buttons'.

ACTIVE MODES

Whilst on the editor screen, you can be in one of two modes of operation, each is denoted by a particular coloured border:-

BLACK border - control refers to the CONTROL PANEL (lower region of screen).

BLUE border - control refers to the SPLIT METER (upper screen region).

To toggle between the two modes press the SPACE BAR.

CONTROL PANEL MODE

The 'function buttons' are operated by positioning a pointer over the relevant function, and selection made by pressing fire. All control is via a joystick in port 2. The available function buttons are;

1 - Increase step in tune (CRSR up key - optional).

2 - Play tune - press space to return from tune.

TUNE FORMAT

A typical TUNE could be of the following form:-

Command/Split to play	Repeat/parameter	Step	<Comment>
SP	08	00	Speed 8
WT	05	01	Delay 05
00	01	02	Play sample split 00 once
02	10	03	Play split 02 16 times
BLANK	00	04	Exits

3 - Decrease step in tune (CRSR down key - optional).
4 - Select area in tune window to edit.
5 - Increase parameter of selected item (; key optional).
6 - Decrease parameter of selected item (= key optional).
7 - Delete highlighted step in tune
8 - Insert before highlighted step in tune.
9 - Exit to file menu.

Note that keeping the space bar depressed (toggling between the two modes of operation), and at the same time pressing the fire button, slows down the effect of a selection. This can be useful if you seem to race through values, and require a 'finer resolution' when selecting.

SPLIT METER MODE

The two sample split indicators, appearing on the meter display (white=start split; red=end split), can be moved by using the joystick in the horizontal, and vertical directions for each indicator respectively. In effect, the sample splits partition the complete sample into 128 different sounds (shown as hexadecimal 00-7F). To select a particular split to edit, use the F1 and F2 keys. Press fire to test the sound split, and once the sound is as you require, store the information by pressing F7. Note that without pressing F7, no new sample split information will be stored. The border will rapidly flash colour to confirm that data for the sample split has been saved. For test purposes only, the speed of the playback can be changed by pressing the F3 and F4 keys. The speeds can range from $00-$FF (0-255), and is the delay between the processing of bits when playing back a sample. Note that a speed of 0 gives the largest delay, being equivilant of 256. The sound quality heard in this test mode will not be as clear as when the tune is re-played, this is because the screen interrupts are not disabled, but will suffice for test purposes.

TUNE FORMAT

The tunes are displayed in the TUNE WINDOW, this is the area to the right of the first block of function buttons. Each step shown in the TUNE WINDOW comprises of; XX YY ZZ; where XX relates to either a value in the range $00-$FF, this being the sample split number to play, OR a command:- SP=select speed to play subsequent sample splits; WT=cause a specified delay before the next step in the tune is processed; RE=command to restart the tune. A blank space in this area signifies the end of the tune. The hexadecimal value referred to by YY can be either the number of repeats of a sample (if a value $00-$7F is specified), or a parameter value to be used with one of the SP/WT commands. Note that both the command RE, and a blank space, do not take the value of YY into account. To make either XX or YY as the field to be edited, select the "< >" function button. (it will toggle between XX and YY). A circle marker will appear above the relevant 'XX' or 'YY' field. When this appears, selecting the "+" and "-" function buttons will increase or decrease the field value. The value ZZ is the step number in the tune.

DEMONSTRATION FILES

I have supplied a demonstration file, and tune/split information. The relevant sound samples are suffixed with ".SAM", tune information with ".TUN", and split information with ".SPL". To listen to the demonstration file, named MISC, enter FILE menu selecting the FILE function button when you wish to load and save data. From the FILE menu;
select option "LOAD SAMPLE FILE" - enter name as "MISC.SAM", select option "LOAD SPLIT DATA" - enter name as "MISC SPL", select option "LOAD TUNE DATA" - enter name as "MISC.TUN". When you have loaded in the required sets of data, select the play function button on the control panel to listen to the demo

PLAYING A TUNE IN YOUR OWN PROGRAMS

Enter the file menu by selecting the FILE button on the editor screen. You will need to save the SAMPLE SPLIT DATA, this contains the start and end addresses for all of the 128 sample splits. Also, a TUNE DATA file is required, this contains all the necessary control codes for speed, delay timings, and a sequence of sample splits to play. Most notably, a copy of the sample file is required to re-play the tune. The quality of sound reproduction is greatly improved by disabling the screen (by clearing BIT 4 of location $D011). To add to this effect, the interrupts are also disabled with SEI throughout the playing of the tune. Please refer to the source file (SAMPLEPLAYER. ASM) for some documentation on memory usage and handling of the tune/sample split data.

Finally, even if you haven't obtained Datel's digital sound sampler to sample your own sounds, I hope you enjoy playing around with the kit - hopefully not annoying anybody in the process.....so turn up the volume!

1581 Internal Disk Commands

Getting to grips with the 1581 disk drive is explained in an easy to understand manner.

PAUL TRAYNOR

1581 Memory

The memory can be split into the following areas;

$0000-$1FFF (#0-#8191)	RAM, which can be split into the following;
$0000-$00FF (#0-#255)	Zero page, job queue, variables.
	The zero page contains many of the important parameters which are required by DOS. The job queue enables the the controller and DOS to communicate with each other.
$0100-$01FF (#256-#511)	Stack, variables, vectors.
	The RAM vector addresses found here are the ones which can be altered to
disable	commands such as scratch or the new command
$0200-$02FF (#512-#767)	Command buffer, tables, variables.
$0300-$03FF (#768-#1023)	Data buffer 0
$0400-$04FF (#1024-#1279)	Data buffer 1
$0500-$05FF (#1280-#1535)	Data buffer 2
	Data buffer 2 is the one where the user commands u3-u8 (or uc-uh) will jump to.
$0600-$06FF (#1536-#1791)	Data buffer 3
$0700-$07FF (#1792-#2047)	Data buffer 4
$0800-$08FF (#2048-#2303)	Data buffer 5
$0900-$09FF (#2304-#2559)	Data buffer 6
$0A00-$0AFF (#2560-#2815)	BAM for first 40 tracks
$0B00-$0BFF (#2816-#3071)	BAM for last 40 tracks
$0C00-$1FFF (#3072-#8191)	Track Cache Buffer
	The track cache buffer helps to increase the speed of input/output operations. This is because all access involves whole physical tracks which are stored in the track cache buffer. If more than one sector is to be read and they are all on the same track then only one disk access will be performed, further sectors will be read from the track cache buffer.
$4000-$5FFF (#16384-#24575	Serial Bus I/O
	I/O is controlled by a 8520A CIA chip
$6000-$7FFF (#24576-#32767)	Floppy Disk Controller (FDC)
	The FDC is a Western Digital 1770 or more recently1772
$8000-$FEFF (#32768-#65279)	32K ROM, DOS and controller routines
$FF00-$FFFF (#65280-#65535)	Jump table, vectors

The Internal Disk Commands allow the user to manipulate the memory of the disk drive. It is also possible to run machine language programs in the memory of the drive. As with Direct Access Commands care must be exercised when using these commands, especially if the user has no or very little experience.

GENERAL RULES

When you are first using Internal Disk Commands you must be aware that any error in programming can lead to the drive hanging up, and the only way to regain control is to switch off and then on again. If a programming error has lead to the drive motor and green light remaining on, then the only way to safely regain control (ie. avoiding corrupting data) would be to reset the computer. Therefore I advise that you should save all programs before running them.

COMMANDS

Memory Read

With the memory read command you can access any of the addresses within the disk drive's memory. Memory-read is the drive's equivalent of the BASIC PEEK function, the command format is shown below.

```
print#15,"m-r"chr$(l)chr$(h)chr$(n)
where;  l=memory address low order
        h=memory address high order
        n=number of bytes
```

The first example program will read any address of disk memory and up to 255 bytes at one time. The address should be entered in decimal form.

```
10    open 15,9,15
20    input"[down]number of bytes
      to read (0=finish)";n
30    if n<1 then close 15:end
40    if n>255 then 20
50    input"starting at address";ad
60    h=int(ad/256):l=ad-h*256
70    print#15,"m-r"chr$(l)chr$(h)
      chr$(n)
80    for j=1 to n
90    get#15,x$:if x$="" then
      x$=chr$(0)
100   print asc(x$);
110   next j
120   print
130   goto 20
```

The second memory-read example program when run will determine the type of disk drive, the device number (x) to be tested is set in line 10.

```
10    x=9
20    open15,x,15
30    open1,x,8,"#1"
40    print#1,"m-r"chr$(198)chr$
      (229)chr$(1)
50    get#1,a$
60    if asc(a$)=0 then dr$="1581":
      else dr$="1571/41"
70    close1
80    close15
90    printdr$
```

" EXERCISE CARE WHEN USING DOS COMMANDS. "

Memory Write

The memory-write command allows you to write to any of addresses within the disk drive's own RAM. Memory-write is the drive's equivalent of the BASIC POKE command, the command format is shown below.

```
print#15,"m-w"chr$(l)chr$(h)chr$
(n)chr$(d1)chr$(d2)....chr$(dn)
where;  l=memory address low order
        h=memory address high order
        n=number of bytes
        d1-dn=data bytes
```

You can write up to 35 consecutive data bytes with one memory-write command.

Our memory-write example program will put a machine language routine into data buffer 2. Operation of this routine is explained later in the text.

```
10    open15,9,15
20    y=0
30    read x:if x=256 then end
40    print#15,"m-w"chr$(0+y)chr$
      (5)chr$(1)chr$(x);
50    y=y+1
60    goto30
70    data 76,7,5,32,229,129,96,
      32,241,129,165,118,197,118,
      240,252,96,256
```

Memory Execute

With the memory-execute command you can execute any routine in drive memory RAM or ROM. Memory Execute is the drive's equivalent of the BASIC SYS command, the command format is shown below.

```
print#15,"m-e"chr$(l)chr$(h)
where;  l=memory address low order
        h=memory address high order
```

To show the memory-execute command at work, load and run the memory-write example program, (note that channel#15 should still be open), and then enter the following command in direct mode

```
print#15,"m-e"chr$(0)chr$(5)
```

This first command will switch the drive activity light on.

This next command will then switch it off

```
print#15,"m-e"chr$(3)chr$(5)
```

Block Execute

The block-execute command is a combination of the direct access command 'u1' and the internal disk command 'm-e'. The data or program in the sector will be read into the buffer and then executed. The command format is shown below.

```
print#15,"b-e:";channel#;drive#;
track#;sector#
print#15,"b-e:channel#,drive#,
track#,sector#"
```

A programming example of the block-execute command is shown in the 1581 user's guide.

USER COMMANOS

User commands tell the 1581 to execute programs starting at certain predefined locations or addresses in it's memory. Below is a table of user commands and there associated functions.

User Command	Function
u0	Restores default user jump table
u1 or ua	Equivalent of direct access command Block-Read
u2 or ub	Equivalent of direct access command Block-Write
u3 or uc	Jump to RAM data buffer #2, Location $0500 (#1280)
u4 or ud	Jump to RAM data buffer #2, Location $0503 (#1283)
u5 or ue	Jump to RAM data buffer #2, Location $0506 (#1286)
u6 or uf	Jump to RAM data buffer #2, Location $0509 (#1289)
u7 or ug	Jump to RAM data buffer #2, Location $050C (#1292)
u8 or uh	Jump to RAM data buffer #2, Location $050F (#1295)
u9 or ui	Jump to the reset tables, Location $FFFA (#65530)
u: or uj	Power up vector

The format for sending a user command is as follows;

```
print#15,"ucharacter";
```

The character is taken from the above table.

Although there is only space for three bytes between each of the u commands (u3-u8) jump-to locations. This is sufficient room for a machine language JMP command allowing you to use longer routines that can be initially called up by 'u' commands. To show an example of the user commands at work load and run the memory-write example program, (note that channel#15 should still be open), you will notice that m-w example program loads the program into data buffer#2 which is the one where the user commands, u3-u8 (uc-uh), jump to therefore this program can be operated by user commands as well as the 'memory-execute' method previously shown. When you have run the 'm-w' example program enter the following command in direct mode.

```
print#15,"u3"
```

This first command will switch the drive activity light on.

This next command will then switch it off.

```
print#15,"u4"
```

UTILITY LOAOER

The utility loader is the command which will load a USR file from disk into disk drive memory where it will then execute. The format for the utility loader command is as follows;

```
print#15,"&0:filename"
```

USER FILES

A user file has to follow certain guidelines. It is limited to just one sector of a disk and this sector is constructed as below;

Byte	
0	Start address low order
1	Start address high order
2	Number of program
3-	Program code bytes
last	checksum

The maximum number of program code bytes is 251. The checksum byte will immediately follow the last program byte. The checksum is calculated by adding all the values of the bytes, starting at the low order start address (byte 0) ,while adding you round down to zero every time you reach 255.

A method of creating user files is to open the file as you would a sequential file (except using ,u,w instead of ,s,w) and then read into this file, from data statements for example, the values of all the bytes starting at byte 0. Below is the framework for a program which will create a user file on disk which can then be subsequently executed by the utility loader command.

```
 10  open2,9,2,"0:&filename,u,w"
 20  read d :if d=256 then 40
 30  print#2, chr$(d); :goto 20
 40  close2:end
 50  data l     :rem start address
                low order
 60  data h     :rem start address
                high order
 70  data n     :rem number of
                program bytes
 80  data
 90  data       :rem program code
100  data
110  data c     :rem checksum
120  data 256   :rem program data
                finished
```

AUTO BOOT LOAOER

When any of the following functions are performed; power up, reset, burst inquire, burst query,or the initialize command, a user type file named "copyright cbm 86", if present, will be loaded into disk memory and executed. After executing the program must call the JCBMBOOTRTN to return control to the drive. there are three methods of disabling the automatic loading of this file;

Setting a flag in the BAM, Byte 7 track 40 sector 1 (there is also a copy in byte 7 track 40 sector 2)

Setting a flag in drive RAM, The JDEJAVU jump table vector mentioned in the user's guide is at location $ff60 (#65376) and that uses the SWITCHAUTO at location $9145 (#37189)

The third method of disabling the auto boot loader is the simplest and that is to just rename the file

ADVENTURE HELPLINE

JASON FINCH **Continues his aid to those Adventurers stuck in the middle of 'KRON'**

Welcome once again to Adventure Helpline. The past three issues worth have concerned themselves with completing the adventure Kron, written by Tony Rome and published by CDU in December 1989.

However, although a solution was given to each problem, the help provided was of a very cryptic nature and not everything was revealed completely. The next three issues (including this one) will virtually duplicate the motions of the first three but this time each of the fifty-four locations will be described in detail and the commands given that you should type in each location to successfully complete the adventure. Not only that, but the results of your actions are also listed, together with all exits and where they lead. I am not going to tell you which ones to take until the end because they will be different if a location is visited twice, once on going somewhere and again on returning from that other location. I shall, however, inform you of directions that you should not take. Suffice it to say that following these and the broad directions given in the other three articles you can't go far wrong. Each location has been given a number, the first being, quite obviously, number one. The layout is quite self-explanatory but for the first location I have provided details of what you need to know. And in this issue stage one, the first twenty-three locations, will be recreated for you. Let the story begin...

1. At the Lagoon of Stars. Rocks surround the beach while to the east stretches the dreaded Sea of Storms. (this is the main description of the location exactly how it is worded on the screen.)

Exits: EAST 2 (this means that by using the verb 'east' you travel to the location numbered two in this series. These exits are not necessarily the only ones because new ones can be discovered by performing various operations)

Type: EXAMINE ROCKS - boat appears (only type what is given in capital letters. The lower case represents the results of the action. This text should only be used on the first entrance to the location. Upon returning just go where you see fit. There may be, like in this case, more than one line of text, so type the first followed by the second and third and so on)

GET BOAT

"ALL THE NICE GIRLS LOVE A SAILOR"

2. On the Sea of Storms east of a cove. The waves scream like angry vultures waiting to devour their prey as they rise and fall in a tortuous rhythm.

Exits: WEST 1

Type: HOIST SAILS - allows south and east movement (to 8 and 3 respectively)

Other info: Do not drop the boat at any stage when you are in the open sea.

3. In open sea. The waves lash mercilessly against the small boat.
Exits: EAST 4, SOUTH 9, WEST 2

4. In open sea. To the south is a sound of rushing water.

Exits: EAST 5, SOUTH 10, WEST 3

Other info: Do not go south

5. On the northeast coast of Sark. To the west is the sea. An old monastery stands on a cliff overlooking the sea. The moon moves behind the dark clouds as they scud across the gloomy sky.

Exits: WEST 4

Type: WAIT - moonlight reveals steps

Other Info: If you have just uncovered the steps, go up them. If coming down, remember to pick your boat up again as it will have been left at the bottom of the cliff.

6. In open sea east of a treacherous rocky inlet off the northwest coast of Sark. To the south shrouded by mist is a strange rock. East is open sea.

Exits: EAST 7, SOUTH 12, WEST 16

Other info: You may here cries here - they come from an eagle elsewhere in the adventure.

7. In open sea.

Exits: EAST 8, SOUTH 13, WEST 6

8. The sea stretches in all directions and bearing south is a tiny island.

Exits: NORTH 2, EAST 9, SOUTH 14, WEST 7

9. In open sea there is a sound of rushing water to the east.

Exits: NORTH 3, EAST 10, SOUTH 15, WEST 8

Other info: The sound of rushing water to the east is in fact a deadly whirlpool. Under no circumstances should move east. It will result in death.

10. The boat is caught in a deadly whirlpool! You are slowly sucked into its spiral path... There is no escape...

Exits: none

Other info: This location represents certain death. Entering here results in you being killed without a chance to escape. Do not enter this location.

11. In an old Boran monastery which is mostly ruins now. The once beautiful stained glass windows are now either cracked or broken. Steps lead down to the shore

Exits: DOWN 5

Type: EXAMINE RUINS - a scroll appears

TAKE SCROLL

Other info: The only way to go is back down the cliff. When at the bottom remember to take your boat again or else you will drown.

12. On the Rock of Akron. A greenish mist surrounds you and voices whisper your name... or is it just the sound of the wind hurrying across the sea...

Exits: EAST 13, NORTH 6

Type: LISTEN - sound comes from a clam

OPEN CLAM - this can only be done if you are holding the bar which is found in location 16. On opening the clam you find an old lamp.

TAKE LAMP

13. In open sea.

Exits: NORTH 7, EAST 14, SOUTH 18, WEST 12

14. On Refuge Rock a small island totally surrounded by the Sea of Storms. A small cave covered with twigs is almost hidden from view

Exits: NORTH 8, EAST 15, SOUTH 19, WEST 13, ENTER 22

Type: GET TWIGS

Other info: If here for the first time, enter the cave by simply typing ENTER CAVE

15. In open sea

Exits: NORTH 9, EAST 15, SOUTH 20, WEST 14

16. On the northwest coast of Sark. The rocks here are jagged and slippy under foot. An old withered tree stands on a ledge a few feet above.

Exits: EAST 6, CLIMB 17 (see below)

Type: GET BAR

THROW ROPE - found in location 22. The rope will wrap around the tree

DROP BOAT - otherwise the branch will snap

CLIMB

"WHO WANTS TO LUG A HEFTY BOAT AROUND?"

17. On a ledge above the rocks. A withered tree stands here, its branches cracked and dry. In the distance is the Castle of Spells towering against the skyline.

Exits: DOWN 16

Type: RELEASE EAGLE - it flies away but leaves a flute

GET FLUTE

GET BRANCH - this will be needed to provide light later on

DOWN

18. At a rocky inlet on the north shore of Sark. Cliffs tower above the waterline... A lone figure stands on the clifftop!

Exits: NORTH 13

19. Off the north shore of Sark. Ahead are the Caves of Goth. It is rumoured that anyone who enters them never returns to tell the tale...

Exits: NORTH 14, SOUTH 23

20. In the Valley of the Storms, a narrow water-way cut through solid rock. A few stones loosen and move as your boat approaches. To the north is open sea and south lies the land you remember so well.. Sark!

Exits: NORTH 15, SOUTH 21

Other info: If you go south from here then you will meet up with death. Do not enter location 21.

21. In a waterway in the Valley of the Stones. Suddenly huge rocks fall from the steep slopes!... They smash your frail boat into small pieces...

Exits: none

Other info: Like location ten, this location represents certain death. There is no escape from the rocks that fall on you. You will be crushed and will die.

22. Inside a small cave on Refuge Rock. This was once a resting place for Pilgrims crossing the Sea of Storms

Exits: OUT 14

Type: GET ROPE

OUT

23. On the north shore of Sark north of the great Caves of Goth, an underground maze of tunnels and pits. Somewhere in the depths of this subterranean world is the Cave of Ice said to hold the key of freedom.

Exits: NORTH 19, SOUTH 24 (covered next month!)

Well that just about wraps up the first stage in detailed terms. To best solve this part you should follow closely the following location motions (eg: 1-4-3-2-5 means start in location one, move to location four and do whatever you are instructed in these instructions, then go to three and do whatever you must do there, then to two, and so on). One of the "correct" methods is as follows: 1-2-3-4-5-11-5-4-3-2-8-14-22-14-8-7-6-16-17-16-6-12-13-14-19-23.

If you follow the movements and do exactly what I have said in each location you should find it possible to move between the locations that I have illustrated and in that order. Just remember NEVER to enter locations 10 or 21 as these will result in you having to start over again. Also, there are twelve locations that represent times when you are in your boat on the waters of the Sea of Storms. Never type DROP BOAT or anything to that effect or you will drown and remember that unless you find land you will die from exposure if you stay in the sea for too long (about thirty moves). Next month we shall have a look at the next seventeen locations which will take you up to the same point as you were at the end of part two of this series. So until then, have fun adventuring...

ADVENTURE WRITING

Creating your own Adventures becomes more and more feasible as this series continues.

JASON FINCH

The past couple of parts of this series have concentrated more on theory and background rather than the actual programming of an adventure. You will have had time to consider what sort of adventure you would like and will have been able to pick and choose what aspects you would like to include. Now though we shall actually get to the main PPPS or Pre-Programming Preparations. "What?! I thought you said we were programming this month!", I hear you shout. Well we will - but only a very small amount. The PPPs must be done so that we can structure the adventure which, in this case, will be programmed mainly in BASIC with a touch of machine code to spice things up a bit. Remember that text compression program that I supplied with the first article? Well the machine code version of the decompression routine is on the disk this month as well as a further two pictures, again created by Doug Sneddon.

These files are "AW-DECOM.MC", "PIC2" and "PIC3" respectively. The assembly language source file is also here for machine code programmers to have a look at, filed as "AW-DECOM.SRC". There are plenty of extra comments to tell you exactly what each step is doing and it is compatible with the 6510+ assembler, published by CDU a little while back. To load the actual code you use the standard ,8,1 suffix and execute the code with SYS49408. Before doing this you must enter POKE2,L where 'L' represents the length in bytes of the compressed data. With this routine you can only have compressed data up to 254 characters in length but an experienced machine language programmer should be able to alter the code to allow more. If you only program in BASIC, never mind.

To see it in action in my AW-COMPRESS program, published with the first article, you should load and run that program as it stands and then press RUN/STOP when it has set itself up with the main screen. Then delete lines 540 to 675 inclusive and replace them with: 540 POKE 2,CL: SYS 49408. Of course, you must have first loaded the code, so delete the command CLR at the very start and replace it with: 5 A=A+1: IF A=1 THEN LOAD"AW-DECOM.MC",8,1. Now you can insert this issue's disk and RUN the program. The code will load and everything will be the same, other than the speed at which the decompressed text is printed!

Right then, a minor change of subject! The very first thing to do when writing an adventure is to come up with an idea for what your adventure will be about. Will it be fantasy or modelled on reality? What will be the setting? - A castle, a fictional world, a house or a futuristic setting are all possible ideas. This is the creative stage. Draw yourself a diagramatic outlook of your adventure landscape. My adventure will be set in a fantasy world and Figure One shows my idea.

This will be a very small adventure but you can see the main components. Once you have got a basic idea of what will happen in

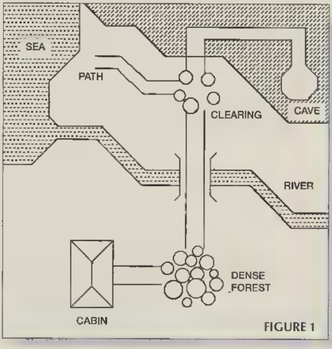


FIGURE 1

your adventure you should create a "square map". This should be much more diagramatic with squares representing individual locations with lines linking them. There are a multitude of different methods of map drawing that you could use but for simplicity and what I call "ease of expansion" use the squares method to begin with. It is a lot easier to add locations than with other methods.

If you want to make an adventure more difficult then make certain "linking paths" one-way. By this I mean that going north to one location does not necessarily mean that by then going south again you return to the same place. By incorporating features like this you can very easily design a "location obstacle". This is something like a maze and in that case you can make all directions possible "exits" but only have one that moves you to a new location. The others return you to the start of the "maze" whilst others could keep you in the same location. By ensuring that the descriptions of these locations are the same, the player can become lost relatively easily. The example adventure, Demad, incorporates a very simply three location maze which is represented by locations nine to eleven in Figure Two, the square map.

A number of terms were used in the last paragraph that I shall now clarify. They are all quite simple but I shall explain them to ensure that you fully understand. Don't forget that this PPP stage is very important. During your programming you must have something to refer to. But first those terms - and a few others.

"Linking paths" are merely the lines that are shown on your square map so that you can easily see which directions are available from a specific location and to which location it leads. The path is not a physical part of your adventure. You could just as easily draw the squares touching with gaps in their sides, but it is not as clear.

"Location obstacles" are features of your adventure that hinder the progress of the player. They are related to the actual location and in which directions the player can leave that location. This could be a maze as mentioned earlier or could be a light/dark situation. This could take the form of a dark tunnel where the player's movement is restricted until a lamp, for example, is lit, whereby new directions are made available.

"Exits" is the term that describes

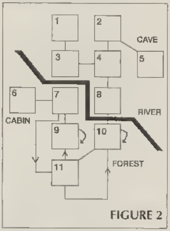


FIGURE 2

which directions a player can move. If on a mountain path that runs north to south, the possible exits are north and south. New exits can be made available under different circumstances. For example, wearing a magic ring could "open up" an exit to the east that may lead into a cave in the mountain.

"Actual obstacles" are things found in the locations, or rooms as I shall now call them. For example if a large boulder prevented you moving west and it was possible to somehow shift it to "open up" the new exit, then this boulder is termed an actual obstacle.

"Objects" are exactly what they say. They are found in the rooms and are the things than can be picked up, dropped, opened, closed and so on. In the previous example the boulder whould also be defined as an object. Some food, a key, a cupboard and a bag of coins are all examples of objects.

Once you have drawn your "square map" and have decided on the general descriptions of the rooms, you should mark on the positions of the actual obstacles. You should have a ratio of about per five to seven rooms. This will require some thought. Next you must decide on how they will be overcome - for example the boulder could be pivotted out of the way using a strong piece of wood. Don't make the solutions too obvious but then again don't make them so outrageously stupid that no-one would be able to solve them. If the latter is the case then the player is likely to quit and put your adventure away to gather dust!

The objects that solve puzzles can be termed **"solution objects"** and all others **"collection objects"**. The latter could be something like a gold coin that is needed to complete the game. Obviously, the solution objects and the actual and location obstacles should be placed with care as thought must be involved to ensure that the solution objects are available before the obstacles are encountered. It would be no good placing the piece of wood to shift the boulder somewhere "after" the boulder. There is quite a bit more to say about solution and collection objects which I shall cover at the start of the next article. For now, though, it's your turn to do a bit of programming!

What we need to design for next month is an input routine. This is very important because it is how the player conveys his commands to the computer and adventure program. You could use a standard INPUT command but a much better one would be specialised to accept only certain characters and length of input. Have a go at that yourself - I shall provide my version next month together with routines to get and display text and graphics and describe the present location. These can be adapted for your own use but will eventually be found in my adventure, Demad. Splitting up the input and analysing it comes later on. Until next month, when we shall have a true programming session, get planning your own adventure!

TECHNO-INFO

Problem solver ***JASON FINCH*** helps a few more readers out.

Dear CDU,
I smoke whilst at the keyboard and this has resulted in my keyboard accumulating ash and going haywire. After dismantling the computer and blowing out the keyboard with a footpump the problem was temporarily rectified. However it occurred again after about a week and required the same treatment. There will come a time when the problem won't be solved in this manner and I wondered if you knew of some supplier who can supply me with a replacement keyboard.
Derrik Nash, The British Forces.

Dear Derrik,
There is a company in Birmingham that will supply replacement keyboards. They are HRS Electronics Plc and their address is Garretts Green Lane, Birmingham, B33 OUE. The telephone number is 021-789-7575 and the keyboards are about forty pounds (including VAT) for a C64 keyboard and around twenty-four pounds for C64C keyboards. However, you will have to contact them for the exact prices. You may, though, be able to get one cheaper by finding someone who has an old 64 that has possibly a number of faulty chips or has been damaged in some other way and is of no use. You could then use the keyboard from that computer or alternatively some computer stores will have ones left from computers that they have been asked to repair but then it has been found that it would be uneconomical. I was able to get hold of one for only five pounds! The best bet though would be to contact HRS.

Dear CDU,
I hope you can help me with the next problem. I want to use a Commodore 8050 dual floppy disk drive with my Commodore 64 computer. The User Manuals says that by acquiring the Commodore 64 IEEE Interface Expansion Card it is possible to attach any IEEE disk drive. So I know that I need an IEEE interface but I cannot find any supplier that stocks these here in Holland. Maybe you can help me to solve my problem. I would appreciate any help or information you can give me that can help me to use my 8050 dual drive.
Hans Sjoerds, Holland.

Dear Hans,
I am pleased to be able to tell you that there is a company here in England that stock the item that you are looking for. They are Meedmore Limited and are based at 28 Farriers Way, Netherton, Merseyside, L39 4XL. The stock number for the Commodore 64 IEEE Interface Expansion Card is A0141 and it costs just short of eighty pounds. This includes VAT and postage in the UK, although I am not sure what arrangements are made and what costs are incurred by sending to other countries. I hope I have been able to set you on your way.

Dear CDU,
I own GEOS which I must say is a brilliant package of programs of which I proudly own most, but is there any way I can have a pound sign. It is very annoying writing a letter to someone when you cannot print a pound sign. I have created my own font with a pound sign but it is not displayed when you press the pound sign key, but when the "at" key is pressed. I have only just found out that this sign appears as a '@' if you print it out using the NLQ mode or Draft mode. Only on High mode will a pound sign appear. Please could you help me.
Matthew Langner, Halfield.

Dear Matthew,
What you must do is alter the appropriate DIP switch on your printer so that the BASIC line OPEN4,4:PRINT#4,"#":CLOSE4 produces a pound sign to be displayed. Then keeping this switch set like that, which I think is the one dealing with international fonts, load up GEOS and type your letter. Whenever you want a pound sign to be displayed use the hash mark (#). Then when you output in NLQ mode, the printer should display a pound sign instead of the hash mark. Thanks go from me to FSSL for providing that bit of information.

Dear CDU,
I am writing to ask if you could solve a little problem I have with a program on the CDU disk Volume 3 Number 8. The file is the POPP program for the C128 computer. The problem is that when I load it into memory I get an Illegal Quantity error in line number ten. I hope you can solve this for me.
A.G.Jones, Hull.

Dear Mr.Jones,
Line number ten of the said program contains only one statement - a WINDOW command. The only possible cause for error that I can imagine is that you are attempting to execute the program in the standard forty column mode. This means that the value for the horizontal width of the window is reduced to a maximum of 39. But the command specifies a width of 79 and therefore, unless you are in eighty column mode, an Illegal Quantity error will be generated.

LETTERS

Dear CDU,
I wonder if you would be good enough to answer the following couple of questions. I have recently acquired a second CBM1541 drive and after a slight repair the machine is as good as new. Is it an advantage and to what uses can I put it?
R.J.Pike, Lichfield.

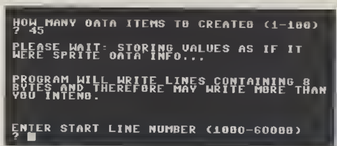
Dear Mr.Pike,
Having a second drive is certainly advantageous even though it may not seem so at the start. This will make itself increasingly apparent as you buy new software or your needs are increased and you will find more and more uses for the new drive. Copying of disks (your own - not piracy, please!) is far quicker using two drives because it eliminates the need to keep swapping the source and destination disks. Also, you have effectively got twice as much space to access at the same time. If using a word-processor or spreadsheet you could have the main system disk in one drive and your work disk in the other. Then if the program needed to keep accessing the main disk you wouldn't need to keep swapping them around. Also there are many uses for a second drive when programming and I am sure that more of them will make themselves apparent as you use it more and time goes on.

Dear CDU,
There appears to be some faults on the CDU disk that I received recently (June issue). None of the 128 programs are found by the drive for a start. Also when trying the Sprite.Bas demo it instructs one to load the Sprite Basic program but pressing the space bar results in the screen going dead, no cursor, no activity. Other programs at fault are the Aleatory Music which, after loading, produces a syntax error in 17190. The copier program is also faulty. After pressing either F5 or F7 there is no activity at all. With regard to the new menu - I do not like it. When using the cursor to select a program it is only possible to go down the program list, using SHIFT does not enable on to go back up the selections again, and I found it only too easy to overshoot the selected program. I have sent the disk back to Select Subscriptions but hope you can shed some extra light on the subject.
E.C.Amesbury, Weston-Super-Mare.

Dear Mr.Amesbury,
It certainly sounds as though the disk was corrupr and I hope that the replacement copy will work when you receive it. However, there may be some orher reason for certain programs not working. The 128 ones for example. These do not load using some 1571 drives because they are protected. This protection must be removed using a disk editor or a directory editor before the programs will load. Alternatively put the drive in 1541 mode. The reason for the Sprite.Bas program not working could be the fault of a fastload cartridge of some description that you have fitted. List line 30 - the last starement should be SYS64738. If it is this then the computer should reset itself. You will have to wait a couple of seconds before the power-up message appears. Aleatory Music can also produce errors if a cartridge is active because of the way it is "decompressed" out to its standard length. If it still produces an error when you get it back, try it without anything else other than the drive and monitor attached. I cannot see what the problem with the menu is. It is programmed ro respond to the SHIFT and cursor down key so that upward movement is possible. These are all only possible causes for the separate errors, of course it may be that parts of the disk have been corrupted in transit.

Dear CDU,
The main reason for writing is to find out how to get multicolour sprites. The user manual does not help much (not at all really). I would also like to know how to get the computer to automatically enter data created from a sprite editor program I have written. I need to enter about ten lines. I hope you can help.
David Mayes, Kent.

Dear David,
First of all, multicolour sprites are controlled by register number 28 in the VIC-II chip, location number 53276. To switch on multicolour mode by entering POKE 53276, PEEK(53276) OR 2^N where 'N' is the number of the sprite from zero to seven inclusive. To switch off multicolour mode use POKE 53276, PEEK(53276) AND 255-(2^N). If you understand binary then the eight sprites are represented by the eight bits of location 53276, sprite zero being the far right bit zero, and sprite seven being the far left bit seven. Thus is you wanted sprites 0,1,4 and 6 to be multicoloured the value to enter would be the decimal equivalent of 01010011 which is 83. Thus you would type POKE 53276,83. As for entering the data, there are a number of ways. By far the simplest is to get the computer to print out a line as if you had typed it, then to print a statement such as GOTO 500 beneath that and then to exit the program. Before doing this the keyboard buffer is loaded with two carriage returns. Then when the program exits, the line will be entered and the GOTO 500 statement executed which returns to the program. However there is one


```
HOW MANY DATA ITEMS TO CREATED (1-100)
? 45

PLEASE WAIT: STORING VALUES AS IF IT
WERE SPRITE DATA INFO...

PROGRAM WILL WRITE LINES CONTAINING 8
BYTES AND THEREFORE MAY WRITE MORE THAN
YOU INTEND.

ENTER START LINE NUMBER (1000-60000)
? █
```

huge problem with this. Whenever a line is entered into the computer's memory, all variables are cleared. If this does not matter then the previous technique is fine. The other method is too complicated to explain here but involves actually playing out with memory contents and pointers. If you want to store a few variables then POKE them to some location (679-767 are handy) before exiting and then retrieve them by PEEKing the same location upon returning. I hope that the technique is suitable because on the disk you will find a program filed as "TECHNO INFO" that will ask you to enter a value between 1 and 100. That many random numbers are then generated and stored in blocks of eight as DATA statements. I have fully REMmed the program so that you can see what is happening. I hope I have been of some assistance.

Dear CDU,

I would like to react to two letters I read in your May 1990 issue. Maybe I can be of some help to the writers. My first remark is for Mr.Booth from Bristol. He experienced trouble with the Power Cartridge. Purely by coincidence I met another C64 user the very day I received CDU who had suffered from similar effects. Here is the story: His C64 (old model) broke down - one of the CIAs blew up, making some operations very hard. So he bought a new one. It worked perfectly until he installed the Power Cartridge. Without it there was no problem. So he switched computers - back to the old one. Result - trouble. He borrowed another 64 from a friend and connected it. Result - system going haywire. In a last attempt he replaced the power supply block of his new 64 with the one of his old 64. Result - all computers worked perfectly (apart from the one with the broken CIA of course). Conclusion - the fault might be in the power supply. To control this he fetched his multimeter and checked out the voltages. And here comes some confusion I don't understand - on the 5 volt supply pin of his oldest power supply (which worked) he found a voltage of 4.65V and on the new supply (the one with the problem) he measured 5.05V. Anyway my conclusion still is that it might be helpful to have your power supply checked out, for this may be the cause of your trouble. My second remark is for Adam Trickett of Leeds. He asked if there is a GEOS printer driver for the Citizen 120D. Well, here is some good news for him: there is such a specific driver, however not in England. I would like to send it to you, but I think that could mean problems with respect to copyrights, for it was published in a German magazine, 64er Sonderheft Nr.28 some time ago. But don't panic: if you write a letter to their publisher, I am sure they would be willing to help you. Their address is: Markt & Technik Verlag AG, Hans Pinselstrasse 2, 8013 Haar bei Munchen, West Germany. Now don't think I am advertising for your "rivals", for 64er publishes lots of good articles and programs and I often buy their magazine at my local newspaper shop, but my postman brings me CDU every month at home, for CDU has the programs on a disk, whereas 64er gives me blue fingers! Keep up the good work!

Jan Kerkhof, Belgium.

Dear Jan,

I cannot thank you enough for all this information and I hope that everything will be sorted out for the people that have experienced problems with BDL's excellent Power Cartridge. I hope Adam can write German! That's a joke of course, I would expect that there is someone there that can read English. Thanks once again for the info.

Dear CDU,

With reference to Simon Searle's letter requesting a copy of Laser BASIC. I have a tape version of White Lightning which contains BASIC LIGHTNING which is almost the same as Laser BASIC which Ocean repackaged. Also included is a version of Forth designed for writing games. All manuals are in a good condition and I would like ten pounds.

Raymond Hoben, Dumbarton, Scotland.

Dear Raymond,

Thanks for your letter. If, Simon, you would like to buy this piece of software from Raymond at a cost of ten pounds then I would like you to get in touch with me again to confirm everything. I will then pass your address on to Raymond and you two can take things from there.

Dear CDU,

Can you please help me. What I would like is for you to ask whether any of your readers has a copy of the Laser BASIC Compiler for the 64. I am willing to pay for it.

D.Beeley, Manchester.

Dear CDU,

With reference to the letter in Techno Info from Adam Trickett in the May 1990 issue, I think the problem may be that the compiler is supposed to compile Laser BASIC and not standard BASIC programs, although I am not sure. I have Laser BASIC and I have been trying to get hold of the compiler for a while so if it is no good to Adam I would like to ask if he will sell it to me (original with instructions please). So if you could please print this letter I would be extremely grateful.

M.Le-Vallois, Paisley, Scotland.

Dear Mr.Beeley/Le-Vallois,

We seem to have a little conflict here. Obviously now, Mr.Beeley, you know of someone with the Laser Compiler who, for one reason or another, was not over impressed with it. Mr.Le-Vallois, you have picked up on a possible cause but of course we do not know whether Adam Trickett really wants to sell this piece of software or not. If both Mr.Beeley and Mr.Le-Vallois would please write to me telling me of the maximum that they would pay or an amount that they would want to pay I would be grateful. Also Adam, could you please get in touch again and tell me whether you are prepared to sell the software, and if so how much you would want for it. In the meantime, anyone else who has a copy of it that they would sell, please write to me with details of how much you would ask for it. I am sure that somehow all this will work itself out and we may even be able to satisfy both Mr.Beeley and Mr.Le-Vallois. I look forward to your further correspondence

TIP OF THE MONTH

Well all my pleas seem to have paid off but we still have only had the following two tips sent to us from you readers. If anyone else has any piece of information that they think may be useful then please send it in to us. The first one is a little batch of POKEs and comes to you courtesy of Mr D.Beeley in Manchester:

POKE775,200 will disable the LIST command, and POKE775,167 will return to normal
POKE774,226:POKE775,252 will result in the computer resetting itself if the LIST command is entered.
POKE649,0 will disable the keyboard, and POKE649,10 will return to normal. This should be used from within a program because otherwise you can't type the second POKE to return to normal!
POKE808,254 will prevent the use of RUN/STOP, and POKE808,237 will return it to normal.

The second tip is slightly longer and more complex and was sent in by Steve Williams, coincidently also from Manchester. It concerns the use of CDU MENU KIT written by Neil Higgins (Volume 3, Number 2):

I have found a way to get the menu as the first file on your disk without the use of a disk editor. Firstly, take a formatted disk and put it in your drive! Then type 10 REM and press RETURN, then type SAVE "NAME",8 where NAME represents the name of the program to go in the menu. Repeat the save instruction as many times as you need, saving each time with a different title for each program. Ensure that the title you give matches the title from the first part of each program (especially if it is in multiple parts). Switch off and then on again before the next bit! Load the CDU MENU KIT and replace the disk with all your titles on. When the program asks if you want to include something you should say yes and include them all. Now sort out your colours and when you have done that save the finished menu to a different newly formatted disk. All you have to do now is save each separate program complete with any extra parts with the file copier from CDU if you feel that way inclined. Then switch off and on again and type LOAD"*",8,1. Your personalised menu is already at the start of your disk-full of goodies. All without the need for a directory editor. It may look a little long winded when you first look at this but it's not when you get stuck in. If you haven't been able to put your menus at the start of your disks before because you didn't have a directory editor then you have no excuse now, have you!

Thanks must go to both Mr.Beeley and Mr.Williams for those tips. Hopefully some more of you will come up with some others now that you see it really is possible to get your tips published! Please everyone remember the NEW ADDRESS if you have any programming problems or general queries, or if you want to have something published in the Tip of the Month section (please mark your envelope with the word TIP). The new address, as published last month, is: **CDU Techno Info, 11 Cook Close, Brownsover, Rugby, Warwickshire, CV21 1NG.** Please note that this is only the address for Techno Info All other correspondence and contributions should be sent to the main address to be found elsewhere in this publication. See you all again next month!

IF AN ADVERT IS IN PRINT, IS IT PROPER?

Most advertisements are perfectly proper
A few are not
The Advertising Standards Authority not only monitors over 850 advertisements every month, it ensures compliance with the rules in the strict Code of Advertising Practice
So when you question an advertiser they have to answer to us
To find out more about the role of the ASA please write to the address below
Advertising Standards Authority, Department X Brook House Torrington Place London WC1E 7HN

ASA

This space is donated in the interests of high standards in advertisements

FEATURE

EXPLORING THE 1541

Learn more about the inner workings of your 1541 disk drive

S WICKHAM

Now that you have purchased your 1541/1570 disk drive,what can you do with it? Well the simple answer is, nothing, until you understand how and why it works. By the end of this artrcle, you should have grasped some knowledge into the inner workings of this 'Rectangular Box'. Hopefully, your usage of the drive will benefit from what you are about to read....

Newcomers to the world of the 1541 will probably only use the drive for storing programs, perhaps they are not aware that you can use the drive for a lot more. The more experienced users will by now be saying to themselves. 'Here we go again, heard it all before'. Before you go rushing off to make a cup of Coffee though, read on....It's never too late to learn new things

This article is MAINLY for the 1541/1570 users, although much of the info is also pertinent to the 1571. Where possible. I will give examples for both units. (For example, everyone is aware that to communicate with the 1541 you use BASIC 2.0 commands, but for the 1571 you can also use BASIC 7 0 commands.) How do you go about learning about something like the 1541, the first thing you should know is how the information is stored on the diskettes that you spend your well earned money on. To be able to understand that, you need to know how a diskette is made up.

Information is stored on the diskette on TRACKS. On a standard 1541 disk there are 35 of these tracks Each track is made up of a number of SECTORS. The sectors are the areas that contain the bytes of data. Each sector holds 256 bytes. The tracks are numbered from the outside to the centre. Therefore, as you get nearer the centre of the diskette, the less number of sectors each track holds. (See 1541 layout). Of these 35 tracks, there's one very important one, this is track 18. Track 18 is known as the BAM(Block allocation map) and and the DIRECTORY track. The BAM shows us what tracks and sectors contain information and which do not, and the Directory track tells us about each file that is stored on the disk (See 1541 layout). Before we go into more detail, opposite is the layout of the tracks, and the sectors of the 1541, together with the sort of information that they contain.

PROGRAM FILE FORMAT

BYTE	DEEINITION
FIRST SECTOR	
0,1	Track and sector of next block in program file 1
2,3	Load address of program
4-255	Next 252 bytes of prg info stored as in comp mem.(keywords tokenized)
REMAINING FULL SECTORS	
0,1	Track and sector of next block in program file1
2-255	Next 254 bytes of prg info stored as in comp mem.(keywords tokenized)
FINAL SECTOR	
0,1	Null ($00), followed by number of valid data bytes in sector
2-???	Last bytes of prg info stored as in comp mem.(keywords tokenized).

The end of a BASIC file is marked by three zero bytes in a row. Any remaining bytes in the sector are garbage and may be ignored.

SEQUENTIAL FILE FORMAT

BYTE	DEFINITION
ALL BUT FINAL SECTOR	
0,1	Track and sector of next sequential data block
2-255	254 bytes of data
FINAL SECTOR	
0,1	Null ($00), followed by number of valid data bytes in sector
2-???	Last bytes of data. Any remaining bytes are garbage & can be ignored

RELATIVE FILE FORMAT

BYTE	DEFINITION
DATA BLOCK	
0,1	Track and sector of next data block
2-255	254 bytes of data. Empty records contain $FF (all binary ones) in the first byte followed by $00 (all binary zeroes) to the end of the record Partially filled records are padded with nulls ($00)
SIDE SECTOR BLOCK	
0-1	Track and sector of next side sector block
2	Side sector number (0-5)
3	Record length
4-5	Track and sector of first side sector (number 0)
6-7	Track and sector of second side sector (number 1)
8-9	Track and sector of third side sector (number 2)
10-11	Track and sector of fourth side sector (number 3)
12-13	Track and sector of fifth side sector (number 4)
14-15	Track and sector of sixth

	side sector (number 5)
16-255	Track and sector pointers to 120 data blocks

DIR FILE FORMAT, TRACK 18 SECTORS 1-19

BYTE	DEFINITION
0,1	Track and sector of next directory block
2-31	File entry 1
34-63	File entry 2
66-95	File entry 3
98-127	File entry 4
130-159	File entry 5
162-191	File entry 6
194-223	File entry 7
226-255	File entry 8

STRUCTURE OF EACH INDIVIDUAL DIRECTORY ENTRY

BYTE	CONTENTS	DEFINITION
0	128+type	File type OR'ed with $80 to indicate properly closed file. (if OR'ed with $C0 instead, file is locked) TYPES: 0 = DELeted 1 = SEQuential 2 = PROGram 3 = USER 4 = RELative
1-2		Track and sector of first data block
3-18		File name padded with shifted spaces
19-20		Rel file only. Track and sector of first side sector
21		Rel file only. Record length
22-25		UNUSED
26-27		Track and sector of replacement file during an @SAVEor@OPEN
28-29		Number of blocks in file, stored as a two-byte integer in normal lo-byte hi-byte format

The above information tells you how

DISKETTE FORMATS & LAYOUTS

Block distribution by Track

Track Numbers		Range of Sectors		Total Sec		S.Sided	D.Sided
HEX	DEC	HEX	DEC	HEX	DEC		
$01-$11	01-17	$00-$14	00-20	$15	21	YES	YES
$12-$18	18-24	$00-$12	00-18	$13	19	YES	YES
$19-$1E	25-30	$00-$11	00-17	$12	18	YES	YES
$1F-$23	31-35	$00-$10	00-16	$11	17	YES	YES
$24-$34	36-52	$00-$14	00-20	$15	21	NO	YES
$35-$3B	53-59	00-$12	00-18	$13	19	NO	YES
$3C-$41	60-65	$00-$11	00-17	$12	18	NO	YES
$42-$46	66-70	$00-$10	00-16	$11	17	NO	YES

BAM FORMAT 1541 - TRACK 18 SECTOR 0

BYTE NUMBER	CONTENTS	DEFINITION
0	18	Track of next directory block. Always 18
1	1	Sector of next directory block. Always 1
2	65	ASCII chararacter A indicating 1541/51/71/4040 format
3		Double sided flag Ignored on 1541
4		Number of sector available on track 1
5		Track 1, sector 0-7 availability map
6		Track 1, sector 8-16 availability map
7		Track 1, sector 17-23 availability map
8		Number of sector available on track 2
9		Track 2, sector 0-7 availability map
10		Track 2, sector 8-16 availability map
11		Track 2, sector 17-23 availability map
	....ETC DOWN TO...	
140		Number of sector available on track 35
141		Track 35, sector 0-7 availability map
142		Track 35, sector 8-16 availability map
143		Track 35, sector 17-23 availability map
144-159	{CHR$(160)}	Disk name padded with shifted spaces
160-161	160	Shifted space {CHR$(160)}
162-163		Disk ID
164	160	Shifted space {CHR$(160)}
165-166		ASCII representation of 2A which are respectively the DOS version (2) format type 1540/41/51/71/4040/2030
167-170		Shifted spaces {CHR$(160)}
171-255		Nulls {CHR$(0)}, not used

1571 DRIVE AS ABOVE EXCEPT:-

3		Double sided flag.$80=Double Sided, $00=Single Sided
171-220		Nulls {CHR$(0)}, not used
221-237		Number of sector available track 36-52 (each sector by each byte)
238	0	Number of sector available track 53 (always 0, all sectors allocated
239-244		Number sector available tk 54-59(each tk by each byte)
245-250		Number sector available tk 60-65(each tk by each byte)
251-255		Number sector available tk 66-70(each tk by each byte)

each track and sector is made up, and what information is contained therein. Later in the article, I will explain just HOW the information is written to the disk Before we get too technical though, I want to show you some of the commands available to you and how we use them. The table below shows you the various commands available, (Using BASIC), both for the 1541/1570 and for the later version 1571. After the table I will demonstrate exactly how to use each one in turn. Using BASIC 2.0 the general format is:- OPEN15,8,15: PRINT#15,"command":CLOSE15 or OPEN15,8,15,"command letter0:information":CLOSE15. (NOTE:- The first 15 in the OPEN/CLOSE command is not mandatory. This is just the file number we allocate to the command. (Normally though 15 is most widely used).

HOUSEKEEPING COMMANDS

BASIC 2.0

NEW	"N0:disk name,disk id"
COPY	"C0:new file=old file"
RENAME	"R0:new nam=old name"
SCRATCH	"S0:file name"
VALIDATE	"V0"
INITIALIZE	"I0"

BASIC 7.0

NEW	HEADER"disk name",id,dv
COPY	COPY"old file"TO"new file"
RENAME	RENAME"old name"TO" new name"
SCRATCH	SCRATCH"file name"
VALIDATE	COLLECT
INITIALIZE	"I0"

FILE COMMANDS

BASIC 2.0

LOAD	LOAD"filename",8 or LOAD"filename",8,1
SAVE	SAVE"filename",8
VERIFY	VERIFY"filename",8
OPEN	OPENfn,8,channel," 0:filename,file type, direction"
CLOSE	CLOSEfn
PRINT#	PRINT#fn,data list
GET#	GET#fn,variable list
INPUT#	INPUTfn,variable list

BASIC 7.0

BLOAD	BLOAD"filename" Bank#,Start address
BSAVE	BSAVE"filename" Bank#,Start address TO end address
BOOT	BOOT"filename"
OPEN	DOPEN#fn, "filename"(record length),(W)
CLOSE	DCLOSE#fn
RECORD	RECORD#fn,record number(,offset)
PRINT#	PRINT#fn,data list
GET#	GET#fn,variable list
INPUT#	INPUT#fn,variable list

DIRECT ACCESS COMMANOS

BLOCK-ALLOCATE	"B-A";0;track; sector
BLOCK-EXECUTE	"B-E";channel, 0;track;sector
BLOCK-FREE	"B-E";0;track; sector
BUFFER-POINTER	"B-P";channel; byte
BLOCK-READ	"U1";channel;0; track;sector
BLOCK-WRITE	"U2";channel;0; track;sector
MEMORY-EXECUTE	"M-E"CHR$ (<address)CHR$ (>address)
MEMORY-READ	"M-R"CHR$ (<address)CHR$ (>address)CHR$ (number of bytes)
MEMORY-WRITE	"M-W"CHR$ (<address)CHR$ (>address)CHR$ (number of bytes) CHR$(data byte) CHR$(data byte) ......etc
USER	"Uchar"
UTILITY LOADER	"&0:file name"
BURST	(1571 only) "U char"+character(s)

Commands intended for the drive are sent over a CHANNEL. Communication with the disk drive can be achieved over any 1 of 15 channels. Channel 15 however is reserved as the COMMAND channel. Data transfer over this channel is as follows:- Opening the channel (OPEN)

Data transfer (PRINT)
Close the channel (CLOSE)

When you initially open the channel, you specify a logical file number, this number must be in the range of 1 to 127, the device number of the drive, (this is normally 8 for single units), and a secondary address. (15 for the command channel. The logical file number is used in any subsequent commands, any number of commands can be sent until the channel is closed. These commands must be referenced by the logical file number first used in the OPEN statement

NEW - Formatting a diskette
The command NEW formats a diskette, that is to say, it prepares a new diskette for receiving data As in all commands, the command word NEW can be reduced to a single letter. EG N=NEW R=RENAME. For clarity, I will show all commands in their condensed format. That is to say that instead of OPEN 15,8,15:PRINT #15,"NEW:name,id". I will use the much shorter method of OPEN15,8,15,"n:name,id". Therefore to Format a new diskette we use the command:-

OPEN15,8,15,"N:name,id"

COPY - Copying files
This command allows the user to copy a file already present on the diskette The command is however seldom used, it's only real benefit is in the ability to combine several SEQUENTIAL files together to make one larger file. This method cannot be employed on PROGRAM files though.

OPEN15,8,15,"C:new file=old file1,old file2,old file 3"

RENAME -.Renames a file with a new name
This command allows the user to change the name of a file on disk. It works on all file types.

```
OPEN15,8,15,"R:new
name=old name"
```

SCRATCH - Scratch a file
This command allows you to get rid of any redundant files. It has the added advantage that you may scratch more than one file at a time.

```
OPEN15,8,15,"S:prog 1"
```
- this would get rid of prog1 only
```
OPEN15,8,15,"S:prog 1,prog
2,prog 3"
```
- this would scratch all 3 files.

(Later on you will learn how you can RECOVER files that have been scratched by mistake)

VALIDATE - Validate diskette
This command allows you to 'Clean up' or Validate your diskette. Whenever you Scratch a program, the program itself is still on the disk. All that happens is that the entry for that program is removed from the directory Validating your diskette makes the space of scratche'd files reusable.

```
OPEN15,8,15,"V"
```

INITIALIZE - Initializing the diskette
The DOS, or Disk operating system, requires a BAM, (Block allocation map), to be present on each disk If you should change disks in the drive when using it, the DOS will not know that you have a different disk in the drive. Therefore it will be working on the old BAM. To combat this, you can initialize the drive. This forces the DOS to read the new BAM

```
OPEN15,8,15,"I"
```

Now that we have dealt with the basic commands for talking to the drive, lets go on to the more exciting commands. These commands are known as the 'Direct Access' commands. Once you understand the concept behind these commands, and what they are capable of, then programming the drive in BASIC is far more entertaining. However, before I go into more detail about these commands, I feel it is time we had a

1541 MEMORY MAP

DRIVE ADDRESSES

HEX	DEC	DESCRIPTION
$0000	0	Command code for buffer 0
$0001	1	Command code for buffer 1
$0002	2	Command code for buffer 2
$0003	3	Command code for buffer 3
$0004	4	Command code for buffer 4
$0006-0007	6-7	Track and sector for buffer 0
$0008-0009	8-9	Track and sector for buffer 1
$000A-000B	10-11	Track and sector for buffer 2
$000C-000D	12-13	Track and sector for buffer 3
$000E-000F	14-15	Track and sector for buffer 4
$0012-0013	18-19	ID for drive 0
$0014-0015	20-21	ID for drive 1
$0016-0017	22-23	ID
$0020-0021	32-33	Flag for head transport
$0030-0031	48-49	Buffer pointer for disk controller
$0039	57	Constant 8, mark for beginning of data block header
$003A	58	Parity for data buffer
$003D	61	Drive number for disk controller
$003F	63	Buffer number for disk controller
$0043	67	Number of sectors per track for formatting
$0047	71	Constant 7, mark for begining of data block header
$0049	73	Stack pointer
$004A	74	Step counter for head transport
$0051	81	Actual track number for formatting
$0069	105	Step size for sector division (10)
$006A	106	Number of read attempts (5)
$006F-0070	111-112	Pointer to address for M and B commands
$0077	119	Device number + $20 (32 dec) for Listen
$0078	120	Device number + $40 (64 dec) for Talk
$0079	121	Flag for listen (I/O)
$007A	122	Flag for talk (I/O)
$007C	124	Flag for ATN from serial bus receiving
$007D	125	Flag for EOI from serial bus
$007F	127	Drive number
$0080	128	Track number
$0081	129	Sector number
$0082	130	Channel number
$0083	131	Secondary address
$0084	132	Secondary address
$0085	133	Data byte
$008B-008D	139-141	Work storage for division
$0094-0095	148-149	Actual buffer pointer
$0099-009A	153-154	Address of buffer 0 $0300
$009B-009C	155-156	Address of buffer 1 $0400
$009D-009E	157-158	Address of buffer 2 $0500
$009F-00A0	159-160	Address of buffer 3 $0600
$00A1-00A2	161-162	Address of buffer 4 $0700
$00A3-00A4	163-164	Pointer to input buffer $0200
$00A5-00A6	165-166	Pointer to buffer error message $02D5
$00B5-00BA	181-186	Record number LO, block number LO

$00BB-00C0	187-192	Record number HI, block number HI
$00C1-00C6	193-198	Write pointer for REL file
$00C7-00CC	199-204	Record length for REL file
$00D4	212	Pointer in record for REL file
$00D5	213	Side sector number
$00D6	214	Pointer to data block in side sector
$00D7	215	Pointer to record In REL file
$00E7	231	File type
$00F9	249	Buffer number
$0100-0145	256-325	Stack
$0200-022B	512-552	Buffer for command string
$024A	586	File type
$0258	600	Record length
$0259	601	Track side-sector
$025A	602	Sector side-sector
$0274	628	Length of input line
$027B	632	Number of file names
$0297	663	File control method
$0280-02B4	640-644	Track of a file
$0285-0289	645-649	Sector of a file
$02D5-02F9	725-761	Buffer for error messages
$02FA-02FC	762-764	Number of free blocks
$0300-03FF	768-1023	Buffer 0
$0400-04FF	1024-1279	Buffer 1
$0500-05FF	1280-1535	Buffer 2
$0600-06FF	1536-1791	Buffer 3
$0700-07FF	1792-2047	Buffer 4

look at the 'Memory Map' of the 1541. To be able to program the drive efficiently, you will need to know it's inner workings better. This is very important once you begin to experiment with M/C programs.

Right now, let's go on to the 'Direct Access Commands'. These commands will all be in BASIC. (Machine Coder's be patient).

Looking at the memory map, you can see that there are 5 buffers. However, only 4 are free for your use. (Buffer 4 is normally used for the BAM). Also please note that when using Seq and Rel files at the same time, buffer 3 is also not available because the Directory uses it. When you wish to use a buffer, you first have to OPEN a channel and specify which buffer you wish to use. For example OPEN 1,8,2,"#2" would open the channel to Buffer number 2. However it is good practice to not specify the actual buffer number but let the DOS select it for you. You achieve this by OPENing x,x,x,"#" If your selected buffer contains Alphanumeric Data, and is not over 88 chars in length. You can use the INPUT# command. (Providing the data is separated by a carriage return). Otherwise you have to use the GET# command. Remember though, that when using GET# it does not allow for null values, therefore we have to check for it via IFA$=""THENA$=CHR$(0).

Before we go any further there are 4 things you must remember:-

1. The PRINT# statement sent to the command channel 15, sends a direct access command to the DOS
2. A PRINT# statement to channels 2 through to 14 sends data to a buffer.
3. An INPUT# or GET# statement to channel 15 returns any error messages.
4. An INPUT# or GET# statement to channels 2 through 14 reads data from a buffer.

The Block-read command tells the 1541 to read a sector from the disk into your openend buffer. (Strictly speaking this is known as a DIRECT ACCESS FILE). Because the first byte of the block does not get read with the Block-read command this command can be shortened to U1 or B-R The Block-write command allows us to copy the buffer contents onto the desired sector on the disk. Block-read can be shortened to B-W or U2 Therefore, the obvious advantage to this command is to READ data into a buffer, alter it, then re-write it back to the disk The Block-Allocate, or B-A command allows the user to reserve blocks on a disk The main purpose of this command is to prevent data from being overwritten. The Block-free or B-F command is the opposite to the B-A command It tells the the BAM which blocks to make available The Buffer-pointer command, shortened to B-P is to tell the DOS just where you wish to start reading or writing data to/from.

The Block-execute, shortened to B-E is quite a powerful command. In essence, you read a sector from the disk into your previously opened buffer. The contents are then executed as a machine code program from within the buffer. In practice when using this command, you specify the buffer number in the OPEN command

Along with the Direct access commands above, you have a few commands that allow you to access the DOS. (Disk Operating System). These are: A.Memory-read B.Memory-write and Memory-execute, shortened to M-R,M-W and M-E respectively

I will now give a few examples of the Direct Access commands in operation. Feel free to experiment, but always make sure that you work on disk with no important data on it. (Mistakes DO happen).

NOTE:- When using the D/A commands, there are two methods available. Either may be used depending upon your own preference:-

Method A is PRINT#15, "U1:"channel number;drive

Method B is PRINT#15,"U1 channel number drive"

If using method B remember to leave a space between each item inside the quotation marks.

BLOCK REAO

Suppose you wished to follow a program through on the disk by track and sector without actually reading the data. To do this you need to follow the path of the 'Link' bytes. That is the 2 bytes at the start of each block that tells you the track and sector of the next block. See Eig. 1.

BUFFER POINTER

Suppose you wish to read the diskette name from within a program As you know the name starts at position 144 of track 18, sector 0. Normally you would have to read the first 143 bytes and ignore them. However the DOS has an easier way. You can point to any position within the buffer by the B-P command The bytes are numbered 0-255 in the buffer, the buffer pointer can be set to zero automatically by the use of the U1 command though. See Fig. 2.

```
1   OPEN8,8,15                                   ;Opens the command
                                                 channel
2   OPEN4,8,4,"#"                                ;Opens the direct access
                                                 file,(no specific buffer)
3   INPUT"Track and sector"                      ,TR,SE
4   PRINT#8,"U1:"4;0;TR;SE                       ;Reads contents ot
                                                 desired Track/Sector into
                                                 buffer
5   GET#4.T$,S$                                  ;Reads the first two bytes
                                                 of the buffer
6   TR=ASC(T$+CHR$(0)):SE=ASC(S$+CHR$(0))        ;Converts string vanable
                                                 to integer, allowing for
                                                 null string
7   IFTR=0THENCLOSE4:CLOSE8·END                  ,If last track then finish
8   PRINT"Track number is· "TR,"Sector number is. "SE ;print them out
9   GOTO4                                        ,Repeat process
```

FIGURE 1

```
1   OPEN8,8,15                          ,Open command channel
2   OPEN4,8,4,"#"                       ,Open direct access file
3   PRINT#8,"U1."4;0;18,0               ,Read contents of desired
                                        Track/sector into buffer
4   PRINT#8,"B·P:"4;144                 ;Point to where we want to start
                                        reading from
5   EORX=1TO16                          ;Length of disk name
6   GET#4,X$.IFX$=CHR$(160)THEN8        ;If shifted space end
7   PRINTX$:NEXT                        ;print out,read next letter
8   CLOSE4                              :CLOSE8:END
```

FIGURE 2

```
1   OPEN8,8,15
2   OPEN4,8,4,"#"
3   PRINT#8,"U1·"4;0;18;0
4   PRINT#8,"B-p:"4,144
5   X$="NEW DISK NAME"
6   IELEN(X$)<16THENX$=X$+CHR$(160):GOTO6
7   PRINT#4,X$,                                  ,Change the contents
                                                 of the buffer
8   PRINT#8,"U2·"4;0;18;0                        ;Write contents back
                                                 to disk
9   PRINT#8,"I":CLOSE4:CLOSE8:END                ;Re-Initialize drive
                                                 and finish
```

FIGURE 3

BLOCK-WRITE

Block-write, is used in conjunction with the block-read command. It allows one to write the contents of a buffer onto the disk at any desired position. The command does NOT alter the contents of the buffer.(You do this task yourself). In the following example we will be changing the disk name that we read with the previous example. See Fig. 3

BLOCK-ALLOCATE

When using Program, Sequential or Relative files on a disk, the BAM is being constantly updated as to blocks that are allocated. This prevents blocks from being overwritten. However, when we use Direct Access files, these are NOT allocated in the BAM, therefore there is a danger that they could be over-written. To prevent this from happening we can use the Block-Allocate command If we try to Allocate a block that has already been allocated, we will be given the error message 65,NO BLOCK,T,S (T and S are the next higher numbered free blocks available).

The syntax for using the Block allocate command Is:- B-A drive track sector The following example would mark track 17 sector 5 as being allocated in the BAM

```
1   OPEN8,8,15
2   PRINT#8,"B-A:"0;17;5
```

BLOCK-FREE

As indicated by it's name, this command frees any allocated blocks and marks them in the BAM as being free to use.

If you wished to make the above track and sector free to use you would use the following

```
    OPEN8,8,15
    PRINT#8,"B-F:"0;17;5
```

NOTE: Allocating and freeing blocks has an effect only on blocks that are used by Prg,seq and rel files by the DOS The B-W and B-R commands do not check the BAM before overwriting blocks. Using these commands you can write to blocks marked as allocated in the BAM. If, for instance, you have a disk that contains only Direct access files, it is unnecessary to allocate written blocks because no other files will be written on the diskette. Therefore in this case you could use the directory blocks in track 18 and therefore have 672 blocks available on the diskette.

To give you an example of the use of this. One could store a menu program onto track 18, thus space on the diskette is not wasted by the menu.

BLOCK-EXECUTE

Block-execute is used when you wish to read a block from the disk into a buffer then execute the contents as a machine code program. The syntax for the command is: B-E channel drive track sector When using the B-E command, the buffer number is usually given in the OPEN command, just in case the M/C prog is not relocatable. IE: OPEN4,8,4,"#2".

```
1  OPEN8,8,15
2  OPEN4,8,4,"#2"
3  PRINT#8,"B-E:"4;0;14;6
```

This would read the contents of track 14, sector 6 The B-E command is used in conjunction with the B-R and Memory Execute commands that follow.

MEMORY COMMANOS

There are three memory commands that we will deal with. They are Memory Read, (M-R) Memory write, (M-W) and Memory execute, (M-E) All these commands pre-supposes are knowledge of the inner workings of the DOS and a knowledge of 6502/6510 code.

The syntax for the Memory read command is:-

```
M-R CHR$(LO) CHR$(HI)
[(CHR$(number)]
```

CHR$(LO) is the low byte of the address in DOS that is to be read.
CHR$(HI) is the high byte of the address in DOS that is to be read
CHR$(number) is the OPTIONAL extra parameter indicating how many bytes to read

In the following two examples, example 1 shows how to read how many free blocks are remaining on the disk. Example 2 shows how to read the disk name

```
1  OPEN8,8,15
2  PRINT#8,"M-R"CHR$(250)
   CHR$(2)
3  GET#8,X$:IFX$=""THENX$=
   CHR$(0)
4  PRINT#8,"M-R"CHR$(252)
   CHR$(2)
5  GET#8,Y$ IFY$=""THENY$=CHR
   $(0)
6  PRINTASC(X$)+256*ASC(Y$)
7  CLOSE8
```

```
1  OPEN8,8,15
2  PRINT#8,"M-R"CHR$(144)CHR$
   (7)CHR$(16)
3  INPUT#8,X$
4  PRINTX$
5  CLOSE8
```

Memory write is the complimentary command to Memory read Writing can only be accomplished to DOS Ram, page zero, stack and the buffers. It is possible to send more than 1 byte with this command. The command syntax is as follows:

```
M-W CHR$(LO) CHR$(HI)
CHR$(NUMBER) CHR$(DATA)
CHR$(DATA) etc etc..
```

Finally, the Memory execute command will call up and execute a machine code program that resides in DOS memory. The routine MUST end with an RTS The syntax for the command is as follows:-

```
M-E CHR$(LO) CHR$(HI)
```

You can not only execute your own routines written with the use of the M-W command, but also the DOS ROM routines.

So now that we have skirted the subject of Direct Access and Memory commands, just what exactly is possible The following table list just a few ideas that readily spring to mind:-

A. You can manipulate the sectors and change the BAM
B. You can make changes to the Directory
C. You can make changes to files
D. You can protect files from accidental erasure
E. You can CLOSE files that are still OPENed
F. You can read and alter any sector that you desire
G. You can prevent directories from being viewed
H. You can prevent directories from being loaded into memory
I. You can recover lost or damaged files
J. You can create data structures that the DOS would not normally recognise
K You could place a menu program within the directory track,thus saving space
L. You could put a simple form of 'Protection' on the disk to prevent illegal pirating of a file.

Really the list is boundless. Only your own imagination will set the limits of what can be achieved by the use of these commands. I cannot stress the importance of making sure you do not use important disks for your experiments.

As you are no doubt aware, the 1541 uses the GCR, (Group Coded Recording), method of storing data onto the disk. If you want to know more about this method, I refer you to 'Your Commodore', Issue JUNE 1986, page 75-77. All I will say on the subject is that by using this method, more information can be stored on the disk than you think is possible.

I hope that this article as given you a better understanding of the 1541, and of how to use it. There are many things that I have left out but these are all covered by the many publications that you can buy There is not enough space here to explain everything in detail. Study the listings of some of the programs in this issued, and of previous issues Practice, Experiment but above all else. .Have fun!!!

DISK·USER

Lineage: 53p per word (+VAT)
Semi display: £11.50 plus VAT per single column centimetre minimum 2cm.
Ring for information on series bookings/discounts.
All advertisements in this section must be prepaid.
Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).
Make cheques payable to Alphavite Publications Ltd.
Send your requirements to,
CLASSIFIED DEPARTMENT
ALPHAVITE PUBLICATIONS LTD., 20 POTTERS LANE, KILN FARM, MILTON KEYNES, MK11 3HF.

(0908) 569819

SOFTWARE

ONLY POOLS AND HORSES

FOOTBALL BOXFORM £14.95 cass £17.95 disk
Not a gimmicky plaything but a genuine statistical analysis of football form the sole objective of which is to improve the chances of winning on the pools or fixed odds. **Written by a pools expert the program has forecast over 50% more draws than would be expected by chance.** Homes, aways & draws are shown in order of merit and true odds given for every match. Merit tables show at a glance the teams currently in form and those having a lean spell. Separate AUSTRALIAN POOLS program included in the price.

RACING BOXFORM & HANDICAP WINNER
£14.95 cass £17.95 disk
Two programmes for the price of one. Boxform weighs up a race by considering the many facets of form. Easily amended to suit the user's ideas. HANDICAP WINNER shows the value for money bets. Over 1000 winners every year and ON DERBY DAY 1990 THE PROGRAM GAVE THE WINNERS OF 7 OUT OF 9 HANDICAPS. 20/1 10/1 7/1 11/2 4/1 4/1 6/4. **PROBABLY THE BEST DAY'S RESULTS EVER ACHIEVED, WITH OR WITHOUT A COMPUTER.**

SPECIAL OFFER: All the above for £24.95 (disks £29.95) plus a FREE program to work out almost any bet. So good it's used by bookies.

Why pay inflated prices? **BOXoft CLEVER** the same – get the **BEST.**
Advertised for five years in the sporting and computer press.
BOXOFT (CDU), 65 Allans Meadow, Neston,
South Wirral, L64 9SQ
Cheque/P.O./Access/Visa Tel: 051-336-2668 (24 hrs)

SOFTWARE

SILVER WING SOFTWARE
THE DEDICATED C64 P.D. LIBRARY
We have the best & latest quality demos utilities & games available
ONLY £2.00 on Tape or Disk
For the best benefits & power from your 64 you can not do without Public Domain Software
For our lists send an SAE to
SILVER WING SOFTWARE
185 CALLOWBROOK LANE,
RUBERY,
BIRMINGHAM
B45 9TG.

SERVICES

COMMODORE SUPPLIES

C64 Power Unit	£21.99
C128 Power Unit	£39.99
C2N/C3N Datasette	£24.99
100 x 3½ Discs	£28.99
C64 Magnum Lightgun	£34.99
C64/128 Serial Cable	£5.99

Prices include VAT & P&P Chq/PO to
OMNIDALE SUPPLIES (CD),
23 Curzon Street, Derby, DE1 2ES
Tel 0332 291219

FULL TERMS AND CONDITIONS ARE AVAILABLE ON REQUEST
TO ADVERTISE RING
(0908) 569819

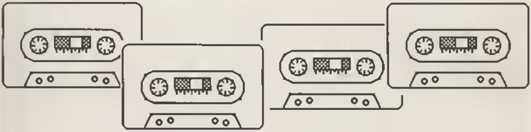
DISK·USER

Lineage **53p** per word (+VAT)
Semi display **£11.50** (+ VAT) per single column centimetre Min size 2cm
Series discounts available on request. All classified advertisements must be pre-paid
There are no reimbursements for cancellations
SEND TO CDU CLASSIFIED DEPARTMENT. ALPHAVITE PUBLICATIONS, 20 POTTERS LANE KILN FARM MILTON KEYNES MK11 3HF

I enclose my cheque/postal order for £ for insertions, made payable to Alphavite Publications.
(Delete as necessary)
PLEASE DEBIT MY ACCESS/VISA CARD NO:
EXP. DATE.
£ FOR INSERTIONS
Name
Address
Postcode
Daytime Tel No
Signature
Date
O FOR SALE O SOFTWARE O SPECIAL OFFERS O REPAIRS O HARDWARE O DISKS

Action, Strategy, Adventure... GAMES!!!

the Great C64 & 128 Cassette Games Offer



Fed up with listings, want a break from programming? Then we've got just the thing for you.
W'ere offering the following games on two cassettes at the ridiculously low price of £3 (for both) including post and packing.

So what's on offer?

Flimbo's quest, Finders keepers, Kentilla, Logic, Microdot, Rainbow chaser, Spots......... all these plus instructions.
Al you have to do is fill in the coupon, (photocopies accepted) and return it to........
ALPHAVITE PUBLICATIONS Ltd, 20 Potters Lane, Kiln Farm, Milton Keynes MK11 3HF.

Please send me 2 cassettes of games.
I enclose my cheque/postal order for £3 made payable to ALPHAVITE PUBLICATIONS LTD.

Or debit my Access/Visa Number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiry Date .... Signature ....

Name .... Address ....

....Postcode..............................

Allow 28 days for delivery MM1

...it's dynamite!

POWER CARTRIDGE

FOR YOUR COMMODORE

64 / 128

SO MUCH FOR SO LITTLE

"...unbelievable value for money" ZZAPP! Dec 89

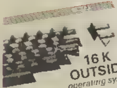
* POWER TOOLKIT
* POWER MONITOR
* TAPE & DISK TURBO
* PRINTERTOOL
* POWER RESET
* TOTAL BACKUP

..."Money well spent" YC/CDU Jan 90

42 Pg Manual - "Damned Good Handbook" CCI Jan 90

AVAILABLE FROM ALL GOOD COMPUTER RETAILERS

TRIED AND TESTED - OVER 100,000 SOLD IN EUROPE



16 K OUTSIDE operating system

..."highly recommended for C64 users" CCI Jan 90

YOU WILL WONDER HOW YOU EVER MANAGED WITHOUT IT

A powerful BASIC Toolkit (Additional helpful commands) that considerably simplifies programming and debugging

AUTO	HARDCAT	RENUMBER
AUDIO	HARDCOPY	REPEAT
COLOR	HEX$	SAFE
DEEK	INFO	TRACE
DELETE	KEY	UNNEW
DOKE	PAUSE	QUIT
DUMP	PLIST	MONITOR
FIND	PLOAD	BLOAD

RENUMBER	Also modifies all the GOTO's GOSUB's etc. Allows part of a program to be renumbered or displaced
PSET	Set up of printer type
HARDCAT	Prints out Directory

The toolkit commands can be used in your programs.

Using POWER CARTRIDGE you can load up to 6 times faster from disk. The Disk commands can be used in your own programs.

DLOAD	DVERIFY	DIR
DSAVE	MERGE	DEVICE
DISK		

MERGE	Two BASIC programs can be merged into one
DISK	With DISK you can send commands directly to your disk

Using POWER CARTRIDGE you can work up to 10 times faster with your data recorder. The Tape commands can be used in your own programs.

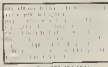
LOAD	SAVE	VERIFY
MERGE	AUDIO	

A powerful machine language monitor that is readily available and leaves all of your Commodore memory available for programming. Also works in BASIC ROM, KERNAL and I/O areas.

A	ASSEMBLE	I	INTERPRET	S	SAVE
C	COMPARE	J	JUMP	T	TRANSFER
D	DIS-ASSEMBLE	L	LOAD	V	VERIFY
F	FILL	M	MEMORY	W	WALK
G	GO	P	PRINT	X	EXIT
H	HUNT	R	REGISTER	$	DIRECTORY
					DOS Commands

The POWER CARTRIDGE contains a very effective Printer Interface that self detects if a printer is connected to the Serial Bus or User Port. It will print all Commodore characters on Epson and compatible printers. The printer interface has a variety of set up possibilities. It can produce HARDCOPY of screens not only on Serial printers (MPS801, 803, 801 etc) but also on Centronic printers (EPSON, STAR, CITIZEN, PANASONIC, etc). The HARDCOPY function automatically distinguishes between HIRES and LORES. Multi-colour graphics are converted into shades of grey. The PSET functions allow you to decide on Large/Small and Normal/Inverse printing. The printer PSET functions are:

PSET 0	Self detection Serial/Centronics
PSET 1	EPSON mode only
PSET 2	SMITH-CORONA mode only
PSET 3	Turns the printing 90 degrees!!
PSET 4	HARDCOPY setting for MPS802/1526
PSET B	Bit-image mode
PSET C	Setting Lower/Upper case and sending Control Codes
PSET T	All characters are printed in an unmodified state
PSET U	Runs a Serial printer and leaves the User port available
PSET Sx	Sets the Secondary address for HARDCOPY with Serial Bus
PSET L1	Adds a line feed, CHR$ (10), after every line
PSET L0	Switches PSET L1 off



On the back of the POWER CARTRIDGE there is a Reset Button. Pressing this button makes a SPECIAL MENU appear on the screen. This function will work with many programmes.

CONTINUE	Allows you to return to your program
BASIC	Return to BASIC
RESET	Normal RESET
TOTAL BACKUP DISK	Saves the contents of the memory onto a Disk. The program can be reloaded later with BLOAD followed by CONTINUE
RESET ALL	RESET of any program
TOTAL BACKUP TAPE	As BACKUP DISK but to TAPE
HARDCOPY	At any moment you can print out a Hardcopy of the screen. Using CONTINUE afterwards you can return to the program
MONITOR	Takes you into the Machine language Monitor

BDL

Bitcon Devices Ltd

88 BEWICK ROAD
GATESHEAD
TYNE AND WEAR
NE8 1RS
ENGLAND

Tel: 091 490 1975 and 490 1919 Fax 091 490 1918
To order: Access/Visa welcome - Cheques or P/O payable to BDL
Price: £16.99 incl. VAT
UK orders add £1.20 post/pack total - £18.19 incl. VAT.
Europe orders add £2.50 Overseas add £3.50
Scandinavian Mail Order and Trade enquiries to: Bhiab Elektronik, Box 216, Norrtalje 76123, SWEDEN Tel: + 46 176 18425 Fax: 176 18401
TRADE AND EXPORT ENQUIRIES WELCOME